

US ARMY RESEARCH OFFICE

Report No. 76-3

September 1976

PROCEEDINGS OF THE 1976 ARMY NUMERICAL  
AND COMPUTERS ANALYSIS CONFERENCE

Sponsored by the Army Mathematics Steering Committee

HOST

TECHNICAL REPORTS SECTION  
STINFO BRANCH  
BLDG. 305

US Army Research Office

Research Triangle Park, North Carolina

11-12 February 1976

Approved for public release; distribution unlimited.  
The findings in this report are not to be construed  
as an official Department of the Army position, un-  
less so designated by other authorized documents.

US Army Research Office

PO Box 12211

Research Triangle Park, North Carolina



## FOREWORD

The first in this series of conferences had as its host the Office of Ordnance Research (now the Army Research Office). It was held in Durham, North Carolina, in late 1962, and was entitled the "ARO Working Group on Computers." It retained this title for only two meetings; then the name was changed to the "Army Numerical Analysis Conference." Recently the name received a modification. The title is now the "Army Numerical Analysis and Computers Conference." This new designation emphasizes both phases of these meetings.

The host for the initial conference also served as host for the present conference--the thirteenth in this series of meetings. The Army Mathematics Steering Committee (AMSC) continues to be their sponsor. Members of this committee would like to thank Dr. Paul Boggs for serving as Chairman on Local Arrangements. He did an outstanding job in carrying out the many tasks associated with conducting a conference of this size.

"The Impact of Mini-Computers and Micro-Processors on Scientific Computation in Army Research and Development" was the theme of the 1976 conference. A Panel Discussion in this area was one of the outstanding features of this meeting. It was chaired by Professor David J. Farber of the University of California at Irvine. The four members of his panel were Dr. E. David Crockett, Hewlett-Packard's Data Systems Division, Professor E. J. Desautels, University of Wisconsin, Dr. Ivan Sutherland, Rand Corporation and Mr. Eric Wolf, Bolt Beranek Newman.

The keynote address was delivered by Dr. E. David Crockett. He titled his talk "Is the Mini-Computer the Next Dinosaur?" Another address which was also closely related to the theme of the conference had as its title the "Evolution of Micro-Computer Technology." It was delivered by Dr. Evan Sutherland. Two other featured speakers were Dr. Achi Brandt, IBM Thomas J. Watson Research Center, and Professor Gene H. Golub, Stanford University. The respective titles for their addresses were "Multi-Level Adaptive Techniques (MLAT) for Discretizing and Solving Partial Differential Boundary Value Problems" and "Least Square and Robust Regression." Members of the AMSC would like to extend their thanks to the above-mentioned panelists and invited speakers for sharing with members of the audience their knowledge about new numerical analysis techniques and new developments in the computer field. Also, they wish to thank those scientists presenting the thirty-three contributed papers. Without their input to this meeting it could not have fulfilled its full significance as an Army conference.

The responsibility for organizing these symposia rests in the hands of the AMSC Subcommittee on Numerical Analysis and Computers. Its chairman, Dr. Ronald P. Uhlig, held an open meeting of this subcommittee on the last day of this symposium. Among the topics brought up for discussion was the theme for the 1977 conference. After considerable interplay of ideas and many suggestions, the theme receiving the strongest indorsements was entitled "Numerical Techniques for Solutions of Nonlinear Partial Differential Equations." We are pleased to be able to announce that the Mathematics Research Center, University of Wisconsin at Madison, Wisconsin, will serve as the host of this coming conference.

# TABLE OF CONTENTS\*

<u>Title</u>	<u>Page</u>
Foreword . . . . .	iii
Table of Contents . . . . .	iv
Agenda . . . . .	vi
A Case History Exploring the Transportability of a Mathematical Algorithm from a Large-scale Computer to a Microcomputer S. Kravitz and J. A. Hauser . . . . .	1
High Speed, Quality Computing on a Minicomputer Edouard J. Desautels . . . . .	21
The Yuma Proving Ground Distributed Computer Ring Network Edward Goldstein . . . . .	29
Linearized Least Squares Larry M. Sturdivan and John W. Jameson . . . . .	49
Nonlinear Spline Regression on Mini-Computers Philip W. Smith and Stanley Hrnccir . . . . .	53
Some Novel Rootfinding Methods Charles E. Gray . . . . .	91
An Improved Iterative Method for Optimizing Symmetric Successive Overrelaxation Vitalius Benokraitis . . . . .	133
VP-Splines, An Extension of Twice Differentiable Interpolation Royce W. Soanes, Jr. . . . .	141
Iterative Solution of the Transonic Potential Equation D. C. Adams and Gary Vander Roest . . . . .	153
Utilizing Real-Time Test Data Analysis in System Monitoring and Checkout E. H. Gamble . . . . .	163
Lethality of a Spectrum of Shaped Charge Projectile Anti-Tank Firepower-Kill Effects Evaluated by the AVVAM-1 Computer Model Donald F. Haskell . . . . .	187

\*This Table of Contents lists only the papers that are published in this technical manual. For a list of the papers presented at the 1976 Army Numerical Analysis and Computers Conference, see a copy of the Agenda.



Development and Optimization of Signal Processing Utilized in a Mine Detection System Abram Leff . . . . .	213
Automated Control, Data Acquisition, and Analyses for Hydraulic Models of Tidal Inlets D. L. Durham; H. C. Greer; III, and R. W. Whalin . . . . .	223
Optimal Instrumentation Planning Using and LDLT Factorization William S. Agee, Robert H. Turner, and Jerry L. Meyer . . . . .	249
A Computer Solution of the Buckingham PI Theorem Using SYMBOLANG, a Symbolic Manipulation Language Morton A. Hirschberg . . . . .	259
PIPS - An Interactive Graphics Program for Determination of Mass Properties of Irregular Planar Solids R. I. Isakower and F. R. Tepper . . . . .	275
Testing Algorithms for a Mini-Computer on a Maxi F. D. Crary . . . . .	333
Use of a Mini-Computer for On-Line Real-Time Processing of Mass Spectral Data from Multiple Mass Spectrometers D. H. Robertson and C. Meritt, Jr. . . . .	343
Recursive Digital Filtering Applied to a Mini-Computer Data Acquisition System Proposed for Army Wind Tunnels R. P. Reklis . . . . .	363
Finite Difference Schemes for Simulating Flow in an Inlet-Wetlands System H. Lee Butler and Donald C. Raney . . . . .	393
Model for Landslide Generated Water Waves Donald C. Raney and H. Lee Butler . . . . .	413
Automatic Euler-Maclaurin Integration Julia H. Gray and L. B. Rall . . . . .	431
Cancellation and Rounding Errors J. Barkley Rosser and J. Michael Yohe . . . . .	445
Applications of Numerical Modeling to Coastal Engineering Problems H. Lee Butler and D. L. Durham . . . . .	471
Automated Data Acquisition and Control Systems for Hydraulic Wave Model Don L. Durham and Homer C. Greer, III . . . . .	509
Attendees . . . . .	521

## A G E N D A

### 1976 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE US Army Research Office Research Triangle Park, North Carolina

All sessions will be held in the Ramada Inn-Downtown, 600 Willard Street, Durham, North Carolina.

#### Wednesday Morning, 11 February 1976

- 0800-0830 REGISTRATION - Triangle Ballroom
- 0830-0845 OPENING OF CONFERENCE - Triangle Ballroom
- WELCOMING REMARKS - COL Lothrop Mittenthal, Commander,  
US Army Research Office, Research Triangle Park, North Carolina
- LOCAL ARRANGEMENTS - Paul Boggs, US Army Research Office,  
Research Triangle Park, North Carolina
- 0845-0945 KEYNOTE ADDRESS - Triangle Ballroom
- CHAIRMAN - Ronald Uhlig, Hqs., US Army Materiel Command,  
Alexandria, Virginia
- SPEAKER - E. David Crockett, Hewlett-Packard, Data Systems,  
Cupertino, California
- TITLE - Is the Mini-Computer the Next Dinosaur?
- 0945-1000 BREAK
- 1000-1215 TECHNICAL SESSION I - Central Carolina Room
- CHAIRMAN- Thomas Dames, Management Information Systems  
Office, US Army Electronics Command, Ft. Monmouth, New Jersey
- A LARGE CAPACITY CIRCUIT ANALYSIS CODE PROGRAMMED ON  
PDP-11/40  
J. C. Ingram, Harry Diamond Laboratories, Adelphi,  
Maryland
- MINI-COMPUTER EXPERIENCE OF A STUDY AGENCY  
Michael E. Gilbertson, Engineer Studies Group, Washington,  
DC

A CASE HISTORY EXPLORING THE TRANSPORTABILITY OF  
MATHEMATICAL ALGORITHM FROM A LARGE-SCALE TO A MICROCOMPUTER  
S. Kravitz, Aeronutronic Ford Corporation, Palo Alto, California

HIGH-SPEED, QUALITY COMPUTING ON A MINI-COMPUTER  
E. J. Desautels, University of Wisconsin-Madison,  
Computer Sciences Department, Madison, Wisconsin

USE OF MICROPROCESSORS AND MINI-COMPUTERS FOR TARGET  
LOCATION

Alan Weinberger and Raymond Coakley, USA Mobility  
Equipment R&D Center, Ft. Belvoir, Virginia

SUMMARY OF PRESENTATION OF THE YUMA PROVING GROUND  
DISTRIBUTED COMPUTER RING NETWORK  
Edward Goldstein, USA Test and Evaluation Command, Aberdeen  
Proving Ground, Maryland

1000-1215 TECHNICAL SESSION II - Duke Room

CHAIRMAN - William S. Agee, National Range Operations  
Directorate, US Army White Sands Missile Range, White  
Sands Missile Range, New Mexico

LINEARIZED LEAST SQUARES  
Larry M. Sturdivan and John W. Jameson, Biomedical Laboratory,  
Edgewood Arsenal, Aberdeen Proving Ground, Maryland

NONLINEAR SPLINE REGRESSION ON MINICOMPUTERS  
Philip W. Smith, Texas A&M University, College of Science,  
College Station, Texas

SOME NOVEL ROOTFINDING METHODS  
Charles E. Gray, Aeronutronic Ford Corporation, Palo Alto,  
California

AN IMPROVED ITERATIVE METHOD FOR OPTIMIZING SYMMETRIC  
SUCCESSIVE OVERRELAXATION  
Vitalius Benokraitis, US Army Ballistic Research Laboratories,  
Aberdeen Proving Ground, Maryland

VP SPLINE, AN EXTENSION OF TWICE DIFFERENTIABLE INTERPOLATION  
Royce Soanes, Computer Science Branch, Benet Weapons  
Laboratories, Watervliet Arsenal, Watervliet, New York

ITERATIVE SOLUTION OF THE TRANSONIC POTENTIAL EQUATION  
D. C. Adams and Gary Vander Roest, US Army Air Mobility  
R&D Laboratory, Ames Directorate, Moffett Field, California

Wednesday Afternoon, 11 February 1976

1215-1315 LUNCHEON

1315-1530 TECHNICAL SESSION III - Central Carolina Room

CHAIRMAN - David Grobstein, Management Information Systems Office, Picatinny Arsenal, Dover, New Jersey

IMPROVED MISSILE SIMULATION DEVELOPMENT USING A HIGHER LEVEL SIMULATION LANGUAGE

Willard M. Holmes, US Army Missile Command, G&C Directorate, Redstone Arsenal, Alabama

UTILIZING REAL-TIME TEST DATA ANALYSIS IN SYSTEM MONITORING AND CHECKOUT

E. H. Gamble, US Army Test and Evaluation Command, Aberdeen Proving Ground, Maryland

LETHALITY OF A SPECTRUM OF SHAPED CHARGE PROJECTILE ANTI-TANK FIREPOWER-KILL EFFECTS EVALUATED BY THE AVVAM-1 COMPUTER MODEL

Donald F. Haskell, US Army Ballistic Research Laboratories, Aberdeen Proving Ground, Maryland

DEVELOPMENT AND OPTIMIZATION OF SIGNAL PROCESSING UTILIZED IN A MINE DETECTION SYSTEM

Abram Leff, US Army Mobility Equipment R&D Center, Ft. Belvoir, Virginia

TRANSIENT THERMAL ANALYSIS OF TRANSCALENT POWER SEMICONDUCTING DEVICES

Russell Eaton, US Army Mobility Equipment R&D Center, Ft. Belvoir, Virginia

1315-1530 TECHNICAL SESSION IV - Duke Room

CHAIRMAN - Sylvan H. Eisman, US Army Frankford Arsenal, Philadelphia, Pennsylvania

FLAT: A FOURIER TRANSFORM

Alfred C. Brandstein, Harry Diamond Laboratories, Adelphi, Maryland

AUTOMATED CONTROL, DATA ACQUISITION AND ANALYSES AND HYDRAULIC MODELS OF TIDAL INLETS

Don L. Durham and Robert W. Whalin, US Army Engineer Waterways Experiment Station, Vicksburg, Mississippi

OPTIMAL INSTRUMENTATION PLANNING USING  $LDL^T$  FACTORIZATION  
William S. Agee, National Range Operations Directorate, US  
Army White Sands Missile Range, White Sands Missile Range,  
New Mexico

A COMPUTER SOLUTION TO THE BUCKINGHAM  $\pi$  THEOREM USING  
SYMBOLANG, A SYMBOLIC MANIPULATION LANGUAGE  
Morton A. Hirschberg, US Army Ballistic Research  
Laboratories, Aberdeen Proving Ground, Maryland

PIPS - AN INTERACTIVE GRAPHIC PROGRAM FOR DETERMINATION  
OF MASS PROPERTIES OF IRREGULARLY SHAPED PLANAR SOLIDS  
Robert I. Isakower and Frederick R. Tepper, Picatinny  
Arsenal, Dover, New Jersey

TESTING ALGORITHMS FOR A MINI COMPUTER ON A MAXI  
Fred Crary, Mathematics Research Center, University of  
Wisconsin-Madison, Madison, Wisconsin

1530-1545 BREAK

1545-1645 GENERAL SESSION I - Triangle Ballroom

CHAIRMAN - Alan S. Galbraith, Durham, North Carolina

SPEAKER - Achi Brandt, Mathematical Sciences, IBM Thomas J.  
Watson Research Center, Yorktown Heights, New York

TITLE - Multi-Level Adaptive Techniques (MLAT) for Discretizing  
and Solving Partial Differential Boundary Value Problems

Wednesday Evening, 11 February 1976

1930-2130 PANEL SESSION ON THEME OF MEETING - Triangle Ballroom

PANEL MODERATOR - David J. Farber, Information and Computer  
Science Department, University of California, Irvine,  
California

PANEL MEMBERS - E. J. Desautels, Computer Sciences Department,  
University of Wisconsin-Madison, Madison, Wisconsin; Ivan  
Sutherland, Rand Corporation, Santa Monica, California;  
E. David Crockett, Hewlett-Packard, Data Systems, Cupertino,  
California; and Eric Wolf, Bolt Beranek Newman, Arlington,  
Virginia

\*\*\*\*\*

Thursday Morning, 12 February 1976

0830-1030

TECHNICAL SESSION V - Central Carolina Room

CHAIRMAN - Stan Taylor, Ballistic Research Laboratories,  
Aberdeen Proving Ground, Maryland

A MINICOMPUTER CONTROLLED BINARY DATA ACQUISITION AND CONTROL  
SYSTEM

J. C. Ingram, Harry Diamond Laboratories, Adelphi, Maryland

USE OF A MINI-COMPUTER FOR ON-LINE REAL-TIME PROCESSING OF  
MASS SPECTRAL DATA FROM MULTIPLE MASS SPECTROMETERS

D. H. Robertson and C. Merritt, Jr., US Army Natick  
Development Center, Food Sciences Laboratory, Natick,  
Massachusetts

ON LINE, ACQUISITION, PROCESSING, AND PLOTTING OF HELMET  
IMPACT TEST DATA

Thomas L. Nichols, US Army Natick Development Center,  
Natick, Massachusetts

RECURSIVE DIGITAL FILTERING APPLIED TO A MINI-COMPUTER DATA  
ACQUISITION SYSTEM PROPOSED FOR ARMY WIND TUNNELS

Robert P. Reklis, US Army Ballistic Research Laboratories,  
Aberdeen Proving Ground, Maryland

APPLICATIONS OF THE MICRO-COMPUTER SYSTEMS FOR IMPROVED  
ARTILLERY TARGET ACQUISITION

Daniel J. Ramer, Picatinny Arsenal, Dover, New Jersey

0830-1030

TECHNICAL SESSION VI - Duke Room

CHAIRMAN - Paul Boggs, US Army Research Office, Research  
Triangle Park, North Carolina

FINITE DIFFERENCE SCHEMES FOR SIMULATING FLOW IN AN INLET-  
WETLANDS SYSTEM

H. Lee Butler and Donald C. Raney, US Army Engineer  
Waterways Experiment Station, Vicksburg, Mississippi

A NUMERICAL MODEL FOR PREDICTING THE EFFECTS OF LANDSLIDE-  
GENERATED WATER WAVES

Donald C. Raney and H. Lee Butler, US Army Engineer  
Waterways Experiment Station, Vicksburg, Mississippi

A NUMERICAL METHOD FOR DETERMINING THE STABILITY CHARACTER-  
ISTICS OF HINGELESS ROTORS

William F. White, Jr., Langley Directorate, US Army Air  
Mobility R&D Laboratory, Hampton, Virginia

AUTOMATIC EULER-MACLAURIN INTEGRATION

L. B. Rall and Julia H. Gray, Mathematics Research Center,  
University of Wisconsin-Madison, Madison, Wisconsin

CANCELLATION AND ROUNDING ERRORS

J. Barkley Rosser, Mathematics Research Center, University of  
Wisconsin-Madison, Madison, Wisconsin

1030-1045 BREAK

1045-1145 GENERAL SESSION II - Triangle Ballroom

CHAIRMAN - J. Barkley Rosser, Mathematics Research Center,  
University of Wisconsin-Madison, Madison, Wisconsin

SPEAKER - Gene H. Golub, Department of Computer Science,  
Stanford University, Stanford California

TITLE - Least Square and Robust Regression

1145-1300 LUNCHEON

Thursday Afternoon, 12 February 1976

1300-1430 OPEN MEETING OF NUMERICAL ANALYSIS SUBCOMMITTEE - Triangle  
Ballroom

---

ADDENDUM

An invited general lecture entitled,  
"Evolution of Micro-Computer Technology"  
will be delivered by Dr. Ivan Sutherland  
of the Rand Corporation, Santa Monica,  
California at 1930 on Wednesday evening.  
This talk will be followed by the panel  
session as announced.





# A CASE HISTORY EXPLORING THE TRANSPORTABILITY OF A MATHEMATICAL ALGORITHM FROM A LARGE-SCALE COMPUTER TO A MICROCOMPUTER

S. Kravitz and J.A. Hauser

Aeronutronic Ford Corporation  
Western Development Laboratories Division  
Palo Alto, California

## Abstract

Problem-solving techniques as adapted to microcomputers compare in varying degrees with those used with large-scale computers and minicomputers. The latest hardware technology has re-exposed some of the same numerical and programming problems that were attendant upon the introduction of minicomputers in replacement of large-scale computers. A case history of a typical microprocessor application is presented, with emphasis on algorithm transportability and microprocessor restrictions and capabilities. The illustrative example is a Graphics Plotter controlled by an Intel-8080 microprocessor. Recommendations for future studies in algorithm standardization are presented.

## A. INTRODUCTION

Technological developments in the field of microprocessors within the past few years have revolutionized the electronics industry. Microprocessor prices have been drastically reduced recently, making it virtually impossible to ignore them as cost-effective alternatives to minicomputers and, in some applications, large-scale computers.

Use of a digital computer for problem solving is typically divided into the formulation, algorithm development, and programming phases. Problem solving can be viewed as a team effort involving a numerical analyst, programmer, and electronic engineer. Microprocessor-based applications generally involve some electronic fabrication which includes circuit design and integration of input/output sensors.

The computer architecture must be taken into account in each development stage. Microprocessor applications differ from large-scale or minicomputer applications in the areas of limited arithmetic capability, lack of adequate support software, and in the requirements of solving real-time data acquisition problems.

The use of large-scale computers for microprocessor software development is an accepted technique. Proofing and simulating algorithms before developing them for the more difficult microprocessor environment is made possible through the use of cross software, such as compilers, assemblers and simulators, which operate on a host computer. However, transporting these algorithms, or those already in use on large-scale computers, still presents numerical and software-language difficulties.

The present day use of large computers to solve engineering problems in general is implemented in high-level languages such as FORTRAN. The use of a high-level language has certain drawbacks, not the least of which is an abstraction (or transparency effect) which masks the nature of the actual computer arithmetic operations. In using microcomputers, however, the practitioner must be keenly aware of the hardware implementation of the arithmetic operations and the numerical problems attendant on small word sizes.

A typical microprocessor system generally consists of the following system components:

**CPU** - The Central Processing Unit controls the communications between memory and the input/output, keeps track of the program, and operates on instructions via the ALU (Arithmetic Logic Unit).

**MCU** - The Memory Control Unit controls which memory chip is accessed by the CPU. A decoder is often used for this purpose.

**DCU** - The Device Control Unit selects the input/output accessed by the CPU. In general, these are the selected port addresses.

**Memory** - Most microprocessors employ both ROM (Read Only Memory) and RAM (Random Access Memory).

**System Clock** - Although some micros now have on-chip clocks, many still require an external clock chip for system timing.

**Interface Chip** - The interface chip is a register (either programmable or not), controlled by the CPU, and is used to interface to the outside world.

**Microprocessor applications** fall into the following general categories: controllers, terminals, communication equipment, and consumer products. Specifically, microprocessors are being used in point-of-sale terminals, onboard vehicle control, banking, recreational games, industrial controls, time-sharing, remote batch, and numerous other applications.

## **B. ALGORITHM TRANSPORTABILITY CONCEPTS**

### **B.1 Algorithm Development**

For this discussion, algorithms are those numerical methods which are used to solve problems on digital computers. By definition, they are completely unambiguous, and should include an error analysis. The error analysis includes accuracy requirements, estimation of round-off and discretization error, step-size and iteration counts, and non-convergence allowances. For the context of this paper, algorithms include programming considerations which are necessary for software implementation.

A much-sought-after objective in scientific computer applications is transportability of algorithms. Transporting an algorithm involves conversion and translation of the computer-dependent features so that the algorithm can be implemented in software on a different computer. Computer algorithms have been developed during the last two decades at great cost and unfortunately, with duplication of effort in conversion to newer and different hardware. Some earlier large computers used for technical problem solving had 36-bit word architecture (with some exceptions, notably the Philco 2000 series, 48-bit word); 32-bit and 60-bit word machines were adapted later. Minicomputers, arriving somewhat later on the scientific-applications scene generally used 16-bit words, although this also varied. More recently, microcomputers became popular. These include 2, 4, 8, 12, and 16-bit architectures. In general, the use of floating-point arithmetic hardware has been confined to large-scale computers and some minicomputers.

With a diversity in number representation, arithmetic capability, and software languages, it is no wonder that the goal of algorithmic transportability seems no closer in the microcomputer era than it did in the minicomputer era. The practical development of a numerical algorithm involves an analysis which considers not only computer word size and arithmetic;

but also vagaries of software languages, software library functions, and, in the case of microcomputers, the peculiarities of real-time data acquisition - notably digitization.

The use of numerical analysis techniques for real-time microcomputer usage differs in some respects from that used in defining a large-scale computer algorithm. Memory and speed constraints restrict the use of extended-word arithmetic and floating-point software. The trade-offs are not always simple nor are the alternatives clearly defined.

The extensive algorithmic and software literature <sup>(1,2,3)</sup> represents a treasure house of ingenious techniques developed with great effort and expense. The future usefulness of microcomputers to solve problems heretofore reserved for larger and more expensive machines, depends upon the conversion techniques adopted. They also depend on the software aids developed to emulate, if necessary, the larger-word machines for which the algorithms were originally developed.

## B.2 Software Development

The general acceptance of FORTRAN as the standard in technical software development has been aided by the efforts of the X3J3 FORTRAN committee <sup>(4)</sup>. Unfortunately for users of mathematical algorithms, standardization is an objective still to be attained. The differences in machine architecture must be taken into account in defining an algorithm. This is especially true in the use of microprocessors. The relative newness of this technology and its application in general problem solving indicates that little in the way of software support tools are available. The first-time user is cautioned against assuming the existence of any extensive mathematical library supplied by the manufacturer.

A purchaser of a microcomputer system can obtain a manufacturer-supplied operating system, assembler programs, and, in some cases, higher-level-language compilers. Unlike the minicomputer environment, suppliers generally do not furnish a comprehensive mathematical library which is fully tested and warranted. However, user groups are being formed, and programs are being exchanged on an informal basis. Faced with the task of implementing an algorithm on a microprocessor, the development team must plan to convert, or transport and implement, existing algorithms (sin, cos, etc.) from larger machines.

High-level-language compilers have been developed as an alternative to machine language. Programs take less time and are easier to write than their assembly language counterparts.

The Intel PL/M cross compiler<sup>(5)</sup> is an early example. Other support programs are being announced in technical publications;<sup>(6)</sup> however, there appears to be little in the way of standardization.

Figure 1 illustrates a typical microprocessor software development cycle. The development team of numerical analyst, programmer, and electronic engineer must interface throughout the cycle to ensure a valid and verifiable implementation.

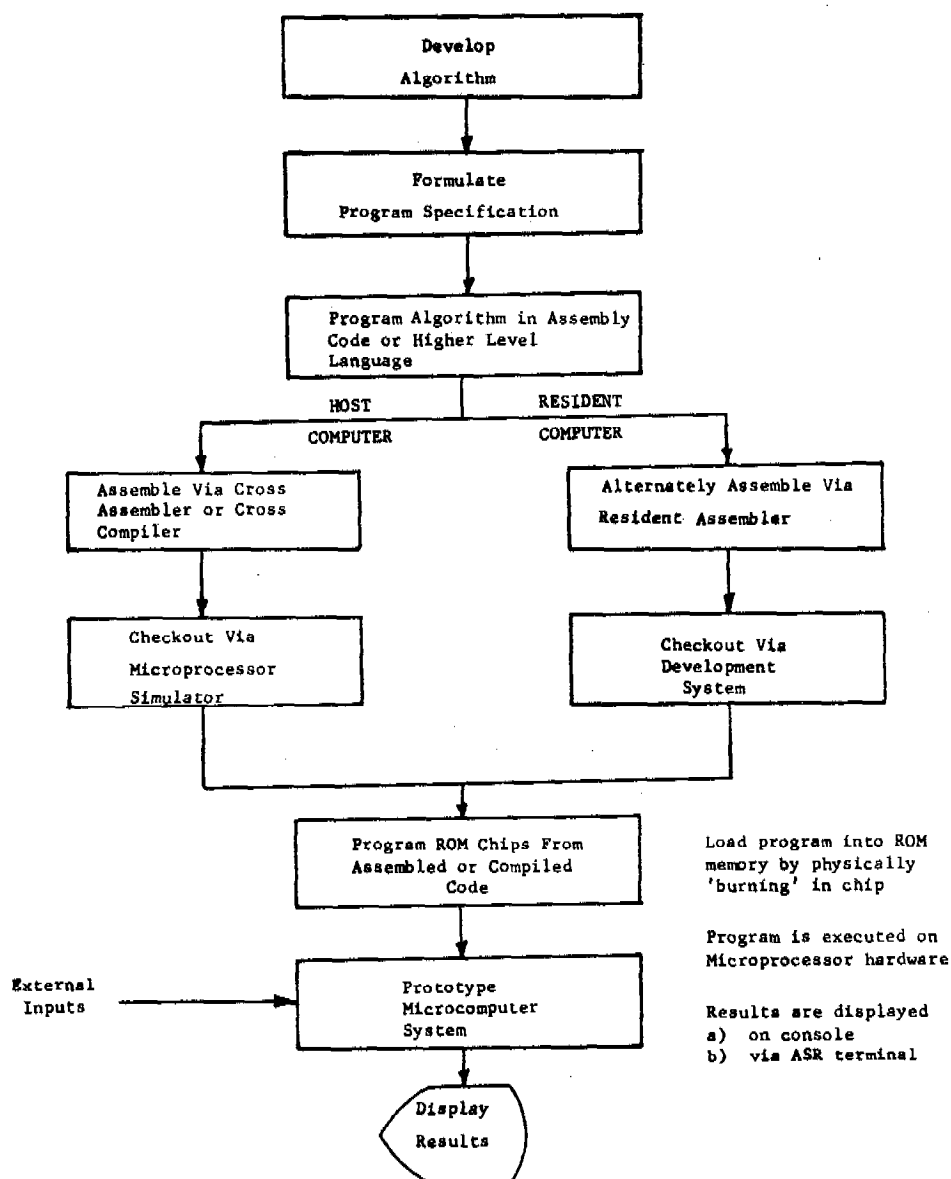


Figure 1. Microprocessor Software Development Cycle

## C. MICROPROCESSOR RESTRICTIONS AND IMPLICATIONS FOR ALGORITHM DEVELOPMENT

### C.1 Hardware Restrictions

Unlike the large scale computer, the microprocessor is restricted in its operation by various inherent parameters. Although nomenclature differences exist between micros, minis and large-scale units, a meaningful comparison can be made.

Word size - The standard microprocessor word lengths vary from 2 bits to 16 bits. However, 8 bits seems to have become the industry standard. Compared to 32-bit or 36-bit words used in many large-scale computers, the limitations are clear. Although multiword definitions increase microprocessor software capability and accuracy, coding, memory, and speed restrictions limit their use.

Speed - The instruction time of microprocessor commands depends primarily on the technology of the unit. Whereas the more popular microprocessors, such as the Intel 8080 and Motorola M6800, use NMOS technology and, therefore, have slow instruction times (approximately 2  $\mu$ s), bipolar microprocessors are becoming more popular and less expensive. These units offer much faster instruction times (approximately 10 to 100 times) at the expense of more power and a larger number of chips per system.

Memory restrictions - The amount of memory a microcomputer system can handle is restricted by the amount of addressing available. Many micros use a 16-bit address bus, enabling a micro to have a capacity of 65K memory. In addition to memory restriction, semiconductor memory access-time restrictions can become important in real-time applications.

Digital to Analog Converter - The restrictions due to Digital-to-Analog Converters (DAC's) are mentioned here because of their implications in the design example. DAC's are specified by resolution (number of bits being converted) as well as type of technology. The higher the resolution the smaller the increment of output voltage each bit represents.

$$V_{\text{out}} = (2^{-n} V_{\text{ref}})$$

where n represents the resolution in bits for unipolar operation.

The design must also take into account the settling time of the DAC. This is the time required for the output function to settle within  $1/2$  LSB for a given digital input stimulus. Various other specifications are important in the determination of the accuracy of the converted number.

## C.2 Software Restrictions

Microprocessor software can be subdivided into two major categories: microprogramming, and fixed instruction set programming. Nearly all bipolar microprocessors are microprogrammable (a user-defined instruction set, utilized at the fundamental register transfer level). Most MOS microprocessors are not microprogrammable and have a fixed instruction set. The more popular Intel 8080 and Motorola M6800 fall into this latter category. Due to general size and speed constraints, microprocessor instruction sets are not as extensive as their large-scale computer counterparts.

Many microprocessor vendors are now making available a higher level language. The more popular ones are PL/M and FORTRAN, with newer ones such as MPL (Motorola) soon to be released. Although these languages alleviate some of the problems of assembly language programming, they create a lot of their own. Some of the problems of assembly language and higher level language are as follows:

Multiply/Divide - Most fixed instruction microprocessors do not have a multiply or divide instruction. Some newer microprocessors consider this problem and either have such instructions, or have a hardware multiply/divide circuit; however, many algorithms have been written to compensate for this oversight.

Fixed Point Arithmetic - Several higher level languages such as PL/M do not incorporate floating-point arithmetic. Although floating-point subroutines have been written and can be called as part of the program, a definite disadvantage both in flexibility and time is recognized.

Unsigned Arithmetic - An annoyance one must keep track of in PL/M is the minus sign in comparisons. In the case of an  $A > B$  check, a negative A which has an absolute value greater than B would appear to be greater than B, whereas it is really less than B.

In real-time data acquisition, the special purpose data gathering hardware is usually designed and constructed simultaneously with the algorithm development. Thus,

simulation tools must be used to verify the algorithms. Especially useful are simulation programs that run on large-scale computers and that allow software to be checked out before final checkout on the candidate hardware. Testing on the candidate hardware will then verify that sensors and conversion devices are correct and adequate. The Intel Intellec microcomputer system permits input/output through a straightforward PL/M command that directly connects the application program to a designated hardware input port. To output a specific value, the following command is used:

OUTPUT (PORT NUMBER) = OUTPUT VALUE

#### D. ILLUSTRATIVE EXAMPLE

The development of a microprocessor-controlled plotter to draw two-dimensional ellipses, although simple, is illustrative of some of the problem areas in algorithm transportability.

##### D.1 Problem Statement

Given the major, minor axis lengths (a,b), draw a smooth ellipse using a microprocessor-controlled analog plotter. This problem is trivial given a modern computer and supporting software. However, as will be shown, the microcomputer solution is not trivial, certainly not for a first-time microprocessor user who has become accustomed to large machine support.

Figure 2 is an artist's rendition of the final objective, a portable microprocessor based, plotting system.

##### D.2 System Design

Figure 3 illustrates the system solution to the ellipse plotter problem. The interactions between algorithm, software, hardware, and human factors, influenced the design.

##### D.3 Hardware

The hardware required for the Graphics Plotter can be divided into two sections: (1) the development hardware, and (2) the prototype hardware. The development hardware consists of a system such as the Intel Intellec 8/Mod 80 with the I/O connected to two 10-bit DAC's



and, subsequently through appropriate amplification, to the X and Y inputs of an X-Y recorder. Data entry is handled through a teletype.

The portable prototype system employs an Intel 8080A microprocessor connected to several 256 X 8 ROM's and 256 X 4 RAM's. In addition, a programmable I/O interface chip handles the input and output, a clock generator and crystal handle the system timing, a 1-of-8 decoder does the device control selection, and a USART is used for data communications through a current loop or RS232 type interface. A system-controller chip handles the proper timing signals to the rest of the system.

The prototype system also employs a keyboard for data entry and an alphanumeric display for information verification and data interchange.

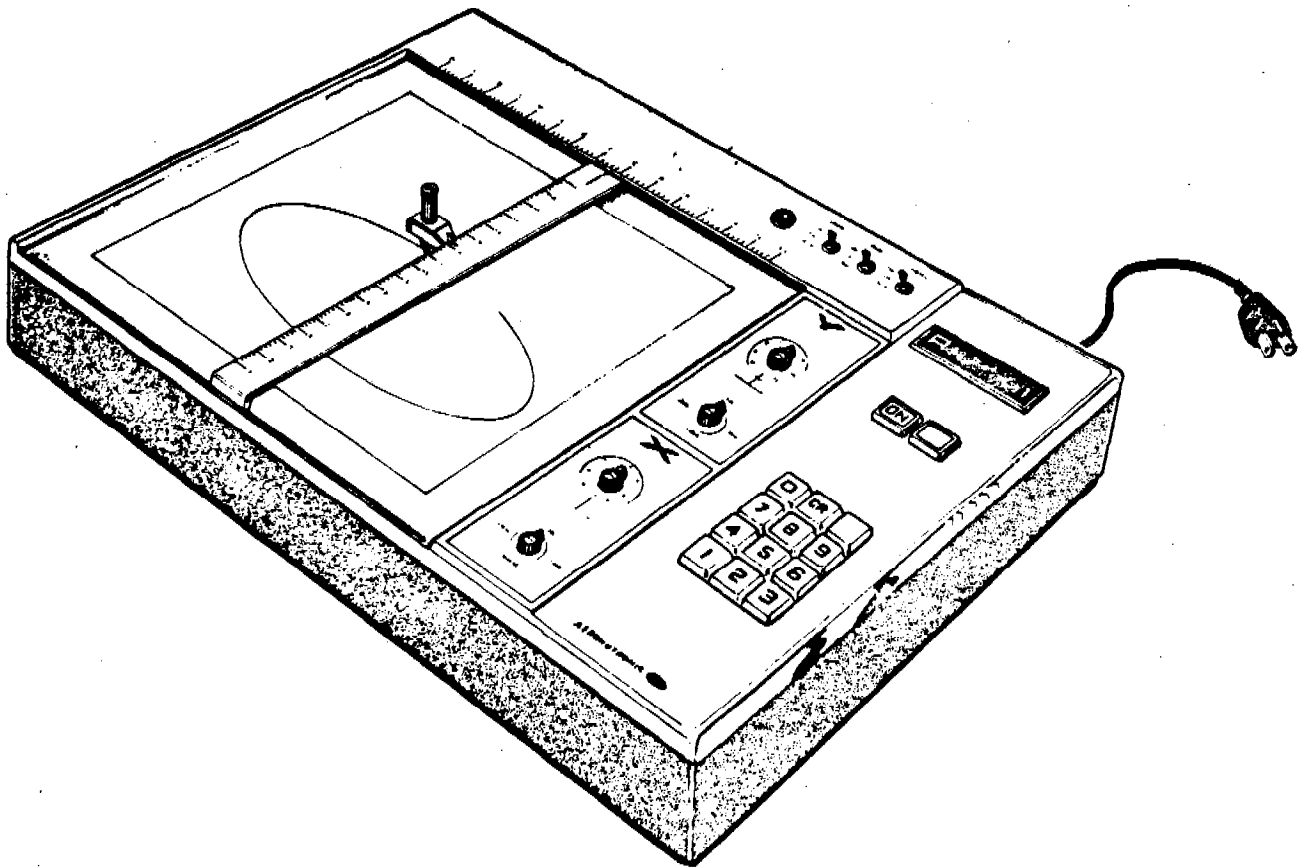
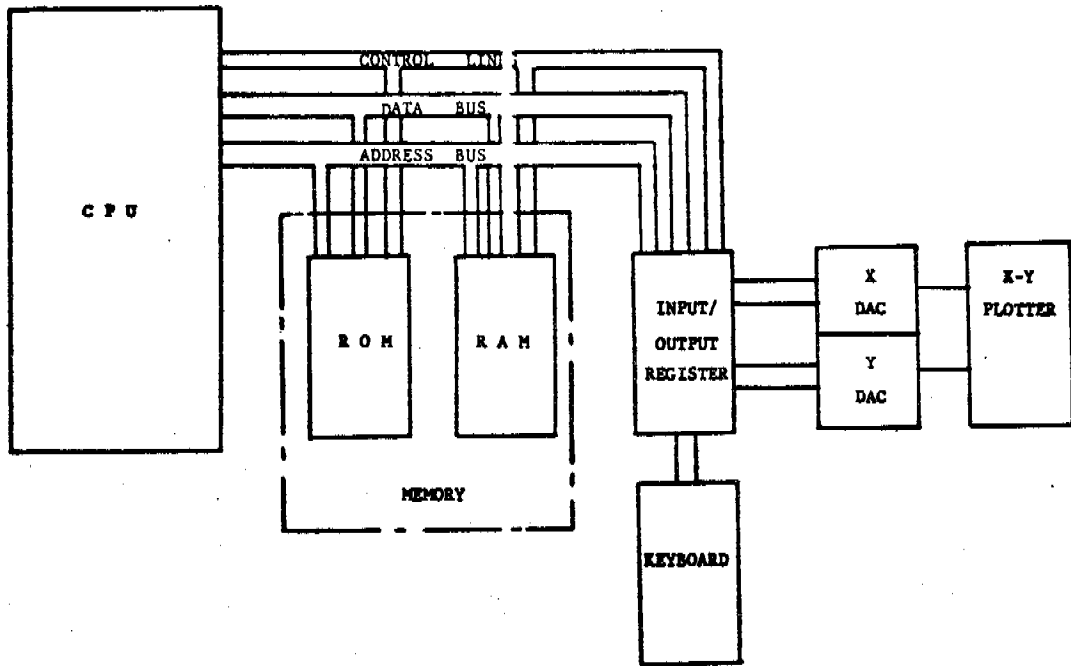


Figure 2. Portable Microprocessor-Based Plotting System



**Figure 3. Microprocessor-Controlled Graphics Plotter-Systems Block Diagram**

#### **D.4 Algorithm**

The following discussion traces the algorithm development in solving the equation of the ellipse on a microprocessor.

##### **Cartesian Approach**

Solve for x and y, given a and b

$$y = (b^2 - \frac{b^2 x^2}{a^2})^{1/2} \quad \text{Eq. D4-1}$$

where x ranges from -b to +b in steps of 0.01 inches

alternatively

$$y = (b - \frac{bx}{a})^{1/2} (b + \frac{bx}{a})^{1/2} \quad \text{Eq. D4-2}$$

## Parametric Approach

$$x = a \cos \theta$$

Eq. D4-3

$$y = b \sin \theta$$

where  $\theta$  ranges from 0 to  $2\pi$  in steps of 0.01 radians.

The cosine function <sup>(7)</sup> was initially approximated by the three-term economized Chebyshev series

$$\cos(x) = 1 + a_2 x^2 + a_4 x^4$$

Eq. D4-4

where

$$a_2 = -0.49670$$

$$a_4 = 0.03705$$

The series was rewritten, replacing the decimal representation with

$$\cos(x) = \frac{1000 - \left( \frac{497 - \frac{K^2}{270}}{2} \right) \cdot \frac{K}{82} \cdot \frac{K}{61}}{1000}$$

Eq. D4-5

where

$$K = x \cdot 100$$

A more accurate six-term series <sup>(7)</sup> currently being implemented is

$$\cos(x) = 1 + x^2 \left\{ a_2 + x^2 \left[ a_4 + x^2 \left( a_6 + x^2 [a_8 + a_{10} x^2] \right) \right] \right\}$$

Eq. D4-6

where

$$a_2 = -0.4999999963$$

$$a_4 = 0.0416666418$$

$$a_6 = -0.0013888397$$

$$a_8 = 0.0000247609$$

$$a_{10} = -0.0000002605$$

The number of multiplications was reduced by factoring, and the decimal constants will be replaced by their reciprocal integer counterparts.

## D.5 Error Analysis

The round-off error<sup>(8)</sup> in solving the equation of the ellipse is represented by the function E.

$$|E| \leq |E_p| + |E_g| \quad \text{Eq. D5-1}$$

where

$E_p$  is the error due to the finite machine representation of real numbers; it includes the propagation error due to the arithmetic operations.  $E_g$  represents the generated error, introduced as a result of imperfect machine arithmetic operations.

An example of generated error is the truncation caused by integer division. In the micro-computer environment, E will depend upon the range of the parameters, the accuracy of the function approximations, word size, scaling, and the fixed or floating-point mathematical operations. Although not shown, a statistical approach can be used to estimate E.

Propagated error may be approximated by Taylor's theorem:

$$\begin{aligned} F(u, v, w, \dots, t) - F(u^*, v^*, w^*, \dots, t^*) &\approx \frac{\partial F}{\partial u} (u - u^*) + \frac{\partial F}{\partial v} (v - v^*) \\ &+ \frac{\partial F}{\partial w} (w - w^*) + \dots + \frac{\partial F}{\partial t} (t - t^*) \end{aligned} \quad \text{Eq. D5-2}$$

where

$u, v, w, \dots, t$  are the true values of the function parameters  
 $u^*, v^*, w^*, \dots, t^*$  are machine approximations

or

$$E_p \approx \Delta F \approx \frac{\partial F}{\partial u} E_{pu} + \frac{\partial F}{\partial v} E_{pv} + \frac{\partial F}{\partial w} E_{pw} + \dots + \frac{\partial F}{\partial t} E_{pt} \quad \text{Eq. D5-3}$$

where

$$\begin{aligned} E_{pu} &= u - u^* \\ E_{pv} &= v - v^* \\ E_{pw} &= w - w^* \\ E_{pt} &= t - t^* \end{aligned}$$

For the Cartesian approach, where

$$F(x, a, b) = \left( b^2 - \frac{b^2 x^2}{a^2} \right)^{1/2} \quad \text{Eq. D5-4}$$

$$E_{py} \approx - \left( \frac{b^2 x}{a^2 y} \right) E_{px} + \frac{b^2 x^2}{a^3 y} E_{pa} + \left( \frac{b - \frac{bx^2}{a^2}}{y} \right) E_{pb} \quad \text{Eq. D5-5}$$

If  $E_{pa}$  and  $E_{pb}$  are assumed to be zero, the Cartesian error may be expressed as:

$$E_{py} \left( \frac{-a^2 y}{b^2} \right) \approx E_{px} x \quad \text{Eq. D5-6}$$

For the parametric approach, where

$$F(a, \theta) = a \cos \theta \quad \text{Eq. D5-7}$$

$$F(b, \theta) = b \sin \theta \quad \text{Eq. D5-8}$$

$\tilde{E}_{px}$  and  $\tilde{E}_{py}$ , the parametric error terms are evaluated as

$$\tilde{E}_{px} \approx \cos \theta E_{pa} - a \sin \theta E_{p\theta} \quad \text{Eq. D5-9}$$

$$\tilde{E}_{py} \approx \sin \theta E_{pb} + b \cos \theta E_{p\theta} \quad \text{Eq. D5-10}$$

Where  $E_{pa}$  and  $E_{pb}$  are assumed to be zero, the parametric error may be expressed as:

$$\tilde{E}_{py} \left( \frac{-a^2 y}{b^2} \right) \approx \tilde{E}_{px} x \quad \text{Eq. D5-11}$$

The Cartesian error,  $E_g$ , is evaluated in functional form as:

$$E_g = f_1 (E_{g1}, E_{g2}, E_{g3}, E_{g4}, E_{g5}, E_{g6}, E_{g7}) \quad \text{Eq. D5-12}$$

where

- $E_{g1}$  = Error introduced by squaring a
- $E_{g2}$  = Error introduced by squaring b
- $E_{g3}$  = Error introduced by squaring x
- $E_{g4}$  = Error introduced by dividing  $b^2$  by  $a^2$
- $E_{g5}$  = Error introduced by multiplying  $\frac{b^2}{a^2}$  by  $x^2$
- $E_{g6}$  = Error introduced by subtracting  $\frac{b^2 x^2}{a^2}$  from  $b^2$
- $E_{g7}$  = Error introduced by the square root function

The parametric generated error  $\widetilde{E}_g$  is evaluated in functional form as

$$\widetilde{E}_g = f_2 (\widetilde{E}_{g1}, \widetilde{E}_{g2}) \quad \text{Eq. D5-13}$$

where

- $\widetilde{E}_{g1}$  = Error introduced by sin or cos series approximation
- $\widetilde{E}_{g2}$  = Error introduced by a or b multiplication

The Intel microcomputer implementation of the Cartesian formulation was made difficult by the following factors:

- a. The limited integer range of 0 to 255 for single-length words and 0 to 65025 for double-length words caused overflow in multiplication, requiring scaling.
- b. Truncation caused by integer division required scaling.
- c. Truncation caused by the integer square root algorithm required scaling.

Extended arithmetic or floating-point arithmetic could have been used to alleviate the above difficulty. This approach is being pursued.

In a similar manner, implementation of the parametric formulation was made difficult by the requirements for an accurate trigonometric approximation and by the scaling required to represent the individual factors and the product of their multiplication.

#### **D.6 Software**

The PL/M language was utilized to implement the ellipse solution<sup>(5)</sup>. The Intel cross-compiler was used on a large scale Honeywell 6060 computer to produce a loadable object paper tape. The simulator program was run later to verify the proper operation of the object program. Using a high level language reduced training and program development time, although the resulting program used more memory resources than anticipated. A flow chart of the program code is shown in Figure 4.

#### **D.7 Problem Areas**

When implemented in PL/M code the Cartesian approach revealed the following problems:

- a. A lack of supplied mathematical library routines. The original square-root function did not satisfy the accuracy requirements.
- b. Division and multiplication worked properly on positive (unsigned) numbers only.
- c. Integer division truncated the results, making accuracy difficult.
- d. Single and double precision word arithmetic was not sufficient to avoid overflow.
- e. The resulting error in the x,y coordinates was too large to present visually smooth ellipses.

The parametric approach solved some of the above problems and the resulting graph was smoother than the Cartesian results. However, a new series expansion for the trigonometric sine function had to be derived.

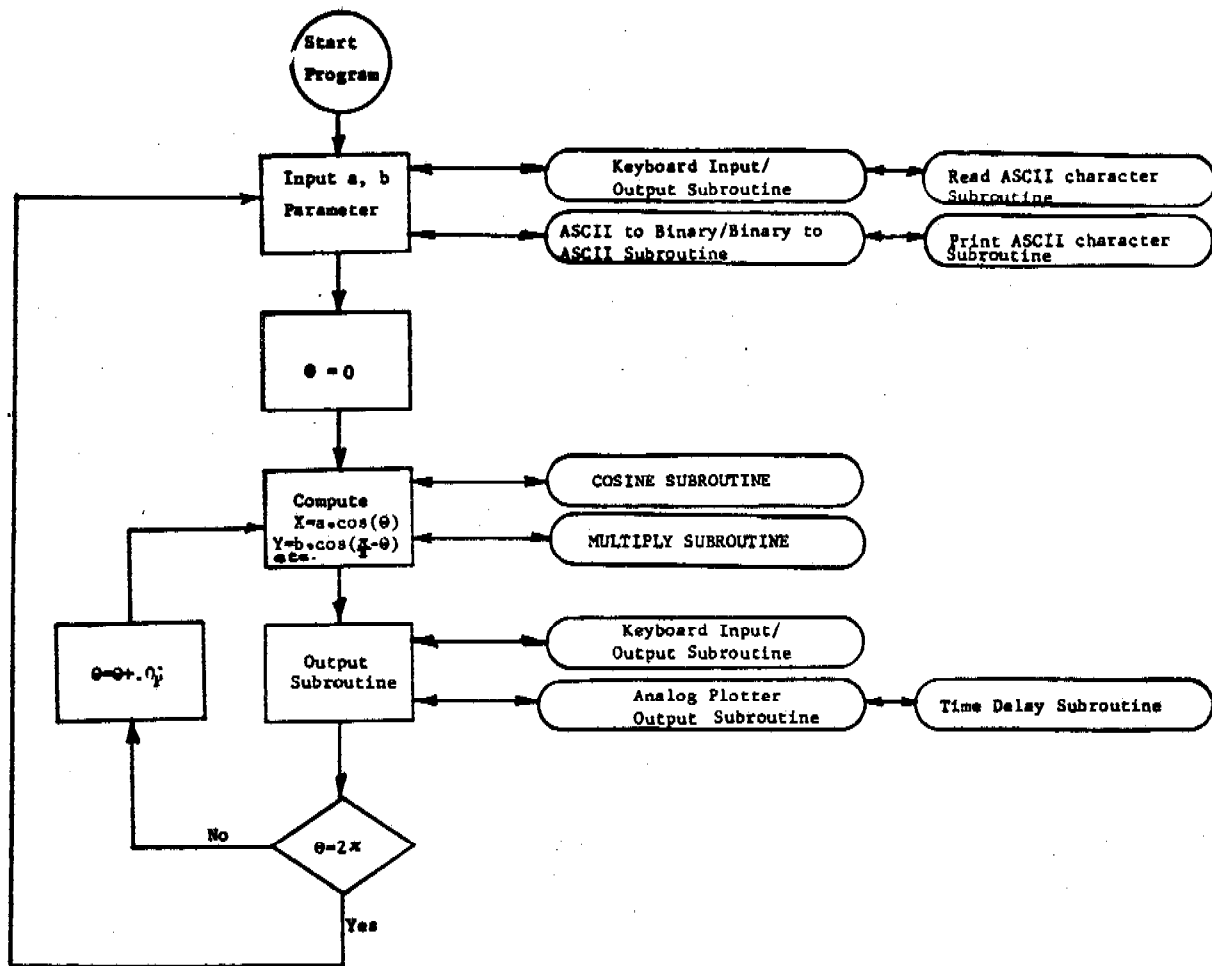


Figure 4. PL/M Program Flowchart for Ellipse

Some of the system and hardware problems were:

- a. The original design used an 8-bit DAC, which was expanded to 10 bits when it was realized that more precision was required.
- b. The classic problem of deciding whether a problem was hardware or software was approached by using the Intellec development system to step through the software as well as by probing the hardware circuitry.



## **E. CONCLUSIONS**

Problem solving with microcomputers is likely to increase in the near future. In some cases, microcomputers will replace minicomputers because of price, size, and low power requirements. To expedite development and reduce redundancy, the following recommendations are made:

- a. Use a development team for microprocessor applications. Include a numerical analyst, programmer, and electronic engineer.
- b. Encourage the development of mathematical libraries, and obtain access to manufacturers' libraries.
- c. Encourage language standardization.
- d. Use host software to check and simulate codes.
- e. Anticipate requirements for multiword and floating-point arithmetic through a thorough error analysis before algorithm implementation.
- f. Debug hardware and software on a development candidate unit such as an Intel MCS or MDS, or a Motorola EXORciser.
- g. Consider the accuracy of the hardware (DAC's, A/D's, etc) before determining software accuracy. There is no need for software to be more accurate than the hardware.
- h. Employ scaling techniques that are appropriate for the individual application.

## **F. IMPLICATIONS FOR ARMY RESEARCH AND DEVELOPMENT**

The acceptance and use of microcomputers by the military will pace usage in the commercial world. High reliability and qualification testing are proceeding, and some acceptance is assumed during 1976.

Microcomputers are viewed not only as a direct replacement for minicomputers but, because of their small size and lower power requirements, as new devices for various tactical applications. The use of minicomputers in radar and fire control systems is already well established. In the future, micros will likely be found on board tanks (automatic turret positioning and fire control, computer-aided fuel injection systems, vehicle communication and display); in guided missiles; in electronic surveillance systems; in navigation aids; and in field-portable communication, process control, and logistics systems. In any instance where digital circuitry is applicable, the use of a microcomputer is suggested, based on lower design costs as well as increased capacity considerations.

The cost factors in using microcomputers are weighed heavily in the man-time costs necessary to prepare software. In a recent article<sup>(9)</sup>, the adoption of standardized processor algorithms to solve problems in electronic warfare is recommended. The author relates the high life cycle costs of custom-designed software as compared to the lower costs incurred by using proven building block modules. Such an approach could result in libraries of algorithms stored on the disks of large-scale computers with support software that would allow the burning in of read-only-memories with those programs of interest. In this style, microcomputer software applications would involve integrating proven modules and preparing only the necessary high-level executive program and if necessary, problem-dependent algorithms.

For future Army application of microprocessors, a study of useful common algorithms and preparation of guidelines for language standardization may be cost-effective in reducing software development and redundancy costs.

## ACKNOWLEDGMENTS

The authors wish to acknowledge the contributions of Messrs F. P. Tatar, and B. G. Stewart.

## REFERENCES

1. Computer Subroutine Libraries in Mathematics and Statistics, International Mathematical and Statistical Libraries, Inc., September 1975.
2. BMD Biomedical Computer Programs, University of California, 1 January 1973
3. COSMIC, Computer Software Management and Information Center, University of Georgia.
4. FORTTRAN Revision, proposed draft ANS X3.9-19XX, American National Standards Institute.
5. Intel 8008 and 8080 PL/M Programming Manual, Revision A, Intel Corporation, 1975.
6. Electronics Design, 20 December 1975 and 5 January 1976.
7. Handbook of Mathematical Functions, ed. M. Abramowitz and L. A. Stegun, U.S. Dept. of Commerce, National Bureau of Standards, May 1968.
8. Elementary Numerical Analysis, S. D. Conte, 1965
9. Standardized EW Software - A Method of Improving Life Cycle Costs, H. I. Hylton, Air Force Electronic Warfare, May/June 1975



## HIGH SPEED, QUALITY COMPUTING ON A MINICOMPUTER

Edouard J. Desautels  
Computer Sciences Department  
University of Wisconsin  
Madison, Wisconsin 53706

ABSTRACT. Factors which should allow the current generation of minicomputers to run large scale scientific computations cost-effectively in comparison to current large systems are discussed. The obstacles to lower-cost software conversion from large systems will decrease due to the advent of minicomputers with large physical memories and large address spaces. Initiation of the development of a numerical analysis problem solving system on a dedicated minicomputer is proposed as a means of exploiting recent hardware and software advances.

1. INTRODUCTION. The title for this paper may be interpreted by some as a statement of the obvious. It is intended to raise the following questions. To what extent can one today conveniently solve large scientific problems quickly and as reliably on minicomputers as on large shared systems such as the CDC 6000, IBM 370/158+ and UNIVAC 1110 series computers? What are the obstacles in reaping the full potential of minicomputers in the mathematical software area? This paper attempts to address these and related questions.

As a preliminary, let us agree that minicomputers for our purposes are general-purpose computer systems deemed sufficiently inexpensive to avoid the need for concurrent sharing via a multiprogramming operating system. Thus a minicomputer would either be dedicated to a single job or it would run a simple batch system, or it would be turned over to single users for hands-on use, etc. Using this definition, what might be considered a minicomputer at one site (e.g. a \$100,000 PDP-11/45) might be used as a shared central facility at another site.

As we proceed with our discussion, the characteristics of minicomputers will also be assumed to change with time. We have to be thinking of the potential uses of minicomputers as they may be 3 to 5 years from now, and what efforts have to be undertaken now in order to be able to fully exploit them as the new generation of minicomputers becomes available.

2. HIGH SPEED. The measures of speed we have in mind are the raw speed usually measured in millions of instructions executed per second (MIPS), sometimes refined as a Gibson mix index (the sum of the products of instruction speeds and instruction frequencies for an assumed characteristic instruction mix). We are also concerned with the perceived speeds as measured by turnaround times under actual operating conditions.

Considering this second aspect first, many times we have heard undocumented allegations as to the overhead associated with large operating systems. Schneck in [9] discusses the factors which can lead to a multiprogramming system consuming an excessive fraction of a system's resources. He advocates monoprogramming as a method of achieving high performance which yields advantages in turnaround time, efficiency and equipment configuration.

Returning to the raw hardware speed, in spite of the small word size of most minicomputers (e.g. 16 bits), single-precision floating point hardware, for 32 bit operands, is now available on many systems for \$5,000 - \$10,000. Many minicomputers have memory cycle times of one microsecond or less. Thus when dealing with memory reference instructions, a rate of 0.5 MIPS or better is attained. Since some minicomputers are equipped with cache memories (a user-transparent high-speed program and data buffer), rates of 1 to 2.5 MIPS are achievable.

In a mathematical problem solving situation, one may wish to distinguish the speeds for two distinct phases of problem-solving. The first phase involves experimentation, program development and trial and error until a suitable approach is found. Interactive computing is the natural mode of computing in this phase. When a suitable approach is found, one enters phase two, the production phase (sometimes called number-crunching).

One can sometimes rather easily cost-justify using a minicomputer for the production phase. For instance, suppose one has access to a shared facility charging \$200 per CPU hour, which typically provides its users with only one tenth of its effective power (because it multiprograms 10 - 15 programs simultaneously to maximize resource utilization). One can imagine a \$100,000 minicomputer system with a speed equivalent to the perceived speed of the large system (a 1 MIPS minicomputer vs a 10 MIPS large computer). The breakeven point for hardware purchase occurs after 500 hours. Since a minicomputer system entails other costs, supporting costs equal to the initial capital investment bring the breakeven point to 1,000 hours (25 weeks at 40 hours/week). Schaefer [8] reports on the use of a minicomputer (a Datacraft 6024/4) for computations in theoretical chemistry as a realistic alternative to machines such as the CDC 7600 or IBM 360/195. He reports that the cost of operating and capitalizing a

minicomputer system can be significantly lower than the cost of operating a very large system which has already been amortized, on the basis of computations per dollar. He concludes that as a theoretical chemist he was able to get three times more computing per dollar using a 1973 vintage minicomputer, and that the technology forecasts point to the price-performance bias towards minicomputers increasing.

Justification for a dedicated minicomputer used for the interactive phase may be more difficult. If one amortizes the equipment over 4 years, then the \$25,000 per year may be equivalent to the cost of a scientific programmer with his overhead. On this basis, the justification may not be so difficult, especially if the system is also used for the production phase, as would be expected. The interactive phase is labor intensive, and any convenient interactive capability suffices, provided it is somehow linked to and software compatible with the production system.

3. QUALITY. Quality mathematical software is being produced by projects such as the the National Activity to Text Software (NATS) [1], and organizations such as the International Mathematical and Statistical Library (IMSL) [3], in addition to efforts put forth by computer equipment manufacturers. As might be expected, these products run on large computers (e.g. CDC 6000/7000 series, UNIVAC 1100 series, IBM 370/360 series, etc.).

Assuming one has access to a high quality library of mathematical software, it is more likely than not written in Fortran, and one has the initial problem of selecting the appropriate subroutines, then mastering the calling sequences and providing for the data handling requirements. This work can be simplified by providing a framework or a coherent working environment for the user.

In attempts to do so, a number of experimental systems were developed in the mid and late 1960's, as described in the Klerer and Reinfelds book [4]. One in particular focussed on providing the user with a means of having a natural notation for problem definition (e.g. mathematical notation), providing for selection of appropriate algorithms (e.g. using polyalgorithms), and using a natural representation of results (e.g. graphical). This was the Numerical Analysis Problem Solving System (NAPSS). The paper by Rice [7] is a retrospective view of the problems and prospects of NAPSS-like systems. Co-existing with the multiprogrammed operating system of a large scale machine was a non-trivial difficulty in the implementation of NAPSS.

NAPSS-like systems do not seem to be available for the current generation of minicomputers. This may in part be due to the fact that it was difficult to fit NAPSS into a system as large as a CDC6500. What efforts might be undertaken at this time?

4. APPROACHES. One can imagine using a minicomputer as a program generator (PG). One would present to the minicomputer a description of a problem in a suitable problem-description-language (PDL) which presumably has the appealing aspects of NAPSS-like languages. The PDL interpreter checks the description for consistency and a limited notion of "correctness". After a dialogue with the problem originator, it generates a series of calls on a mathematical software library, and transmits this collection to a large system, as a remotely-submitted batch job.

The program generator approach is attractive, but it probably is the least effective use of a dedicated minicomputer system. It is an attractive application for a minicomputer-based timesharing system, but it may be equally cost-effective on a large-scale system.

The second approach we will discuss assumes that, if it is almost cost-effective to perform both phases of problem solving on a minicomputer today, it will be more so within three to five years. The second approach involves dedicating a minicomputer to the support of a NAPSS-like system. Instead of having to co-exist with a general purpose multiprogram operating system, we can assume the operating system will be designed to meet the needs of its one and only user. Thus it does not suffer the depletion of resources which seems to characterize multiprogram resource sharing systems, it does not have to contend with protection problems either for security or as protection against unreliable concurrent users. Nor does it need expend much time in detailed cost accounting for resource utilization.

Of course we realize that some protection services are useful as debugging and program checkout tools, and that some accounting information can be used to identify performance bottlenecks. However, instead of supporting services for the primary benefit of the system (e.g. protection, accounting), we would prefer supporting services of direct benefit to the user (e.g. debugging, performance measurements).

5. CURRENT LIMITATIONS. With a few exceptions, most current minicomputers are restricted to directly addressing 64KB (8 bit bytes). In a mathematical software context, this would provide for a maximum of 16K floating point 32 bit words, or a square matrix of 126 by 126. In most systems this would exhaust all of memory. In some, another 32KW (16 bits) would be available for program storage. In a few systems, one could have perhaps 1MB of physical memory, while still being restricted to the small address space described above. The report by Poppendieck and Desautels [6] describes the range of memories and restrictions available on current minicomputers, and their implications.



As mentioned previously, floating point arithmetic of suitable speed and precision is now available on many minicomputers. The availability of sufficient mass storage (e.g. disk) used to be a problem with minicomputers. Fortunately relatively inexpensive drives (80 MB) are available for under \$20,000.

One additional consequence of operating a dedicated system is the possibility of exploiting the microprogramming option which is available on some minicomputers. It is very difficult to support user microprogramming on a shared system, and it is simpler on a dedicated system. In some cases this can yield a performance increase of a factor of 2-5 or better.

6. A POSSIBLE APPROACH. The mathematical software written for large computers might be adapted for use on the current generation of minicomputers. Much of this software is written in Fortran, and adaption might appear to be straightforward. However since most current minicomputers have difficulty handling even 16K floating point operands, much of the code would have to be rewritten to implement in software the virtual memory hardware support for large direct addressing found on some large computers. Even after an expensive adaption process, performance would leave much to be desired.

Recognizing that some minicomputers now support direct addressing spaces comparable to current large computers, and assuming that technological forecasts indicating the cost of logic and memory decreasing by a factor of two every two to three years, we would propose to initiate development of a NAPSS-like system on a dedicated minicomputer with a direct addressing capability of at least 1 MB (20 bits). This is equivalent to a floating point word capacity of 256 KW (32 bit words) which exceeds the direct addressing capability of large systems such as the CDC 6000 series and UNIVAC 1100 series (restricted to 65KW each). By the time efforts to convert large machine code into 16-bit minicomputers might begin to bear fruit, minicomputer technology is likely to have advanced to the point where large memories and large direct addresses would have invalidated most of the conversion effort. The recent interview on the design of the PDP-11 is illuminating in this respect [10].

One can argue that the benefits of improved technology apply to large and small systems equally, yet it appears to be the case that the smaller systems increase in capability while decreasing in price much more rapidly than the large systems.

A very attractive development approach would involve beginning software development on a minicomputer which has the appearance from the programming viewpoint of having a 32 bit word. The Interdata 7/32 has the logical characteristics of a 32 bit-word computer, but it

provides this at a low cost by using 16 bit internal data paths. Having developed software on this "slow" system, one can then upgrade into a faster system, the Interdata 8/32, which is upward compatible with the 7/32. The 8/32 provides 32 bit data paths and other enhancements such as an instruction cache, so that its performance is claimed to equal that of an IBM 370/158.

A computer such as the 8/32 is attractive in that it also supports a writeable-control store, with which one can write micro-code tuned to support the current application. This can provide performance increases and opportunities for monitoring arithmetic errors.

Further performance increases can be obtained through the addition of outboard arithmetic units such as the one manufactured by Floating Point Systems [2]. In principle it is capable of a maximum of 12 million floating point instructions per second, on 38 bit operands. The problem of course is how to keep it busy [5].

7. PRACTICAL CONSIDERATIONS. Developing software is an expensive activity, and it is not likely to decrease in cost. The prospects for transporting software ("software portability") with little effort are not too promising for mathematical software, because of the consequences of minute differences in arithmetic, as well as the usual difficulties.

Since the cost of main storage continues to decrease, it would seem foolish to expend much software conversion effort to adaptation of quality mathematical software developed for large machines so that it can run on minicomputers with minute direct addressing capabilities.

8. CONCLUSIONS AND RECOMMENDATIONS. Current minicomputers are in some instances as cost-effective for scientific computations as large scale systems. Within the next few years, minicomputers are likely to be much more cost-effective, provided one finds a way to minimize the cost of adapting quality mathematical software which has been developed for large scale systems.

Such adaptations should be performed for the newer generation of minicomputers with larger direct addressing capabilities (1 MB or greater), so that conversion costs remain reasonable.

#### REFERENCES

1. CODY, W. J. The FUNPACK package of special function subroutines. ACM Trans on Math Software 1, 1 (March 1975), 13-25.
2. Floating Point Systems, Inc., Portland, Oregon
3. International Mathematical and Statistical Libraries, Inc., Houston, Texas.

4. KLERER, M. and J. REINFELDS, eds. Interactive systems for experimental applied mathematics, Academic Press, New York, 1968.
5. LYNCH, W. C. How to stuff an array processor. Proc. of Third Texas Conf. on Computing Systems, Nov. 7-8, 1974, pp. 3-2-1-3-2-2.
6. POPPENDIECK, M. and E. J. DESAUTELS. Memory extension techniques for mini-computers. Univ. of Wisconsin, Computer Sciences TR #270, March 1976.
7. RICE, J. R. NAPSS-like systems - problems and prospects. Proc. AFIPS NCC Conf., 1973, pp. 43-47.
8. SCHAEFER III, H. F. Are minicomputers suitable for large scale scientific computation? Proc. IEEE Compcon, Sept. 1975, pp. 61-63.
9. SCHNECK, P. B. The myth of multiprogramming. Software-Practice and Experience, 4 (1974), pp. 59-62.
10. SCRUPSKI, S. E. Designing the PDP-11: trials, triumphs. Electronics, Jan. 22, 1976, pp. 75-76.



# THE YUMA PROVING GROUND DISTRIBUTED COMPUTER RING NETWORK

Edward Goldstein  
U. S. Army Test & Evaluation Command  
Directorate for Management Information Systems  
Aberdeen Proving Ground, Maryland 21005

## ABSTRACT.

Most present day real-time data systems at test ranges have been implemented using large scale computers such as the UNIVAC 1108's at White Sands Missile Range. In selecting a new real-time system for Yuma Proving Ground (YPG), the US Army Test and Evaluation Command is applying a new approach - Think Small! With the increasing sophistication and capability of minicomputers rapidly approaching those of large scale computers, it is possible, by separating the various processing elements used in a real-time system and substituting interconnected minicomputers, to do the same job for much less cost. This is the Distributed Computing System approach.

Several distributed computer configurations now in existence are examined in this presentation as well as the YPG proposed system. Included are the Carnegie-Mellon Multi-Processor (C.mmp) System, the Bell Labs Distributed Network and the University of California at Irvine (UCI) System.

## INTRODUCTION.

The paper I am to deliver covers in part a subject on which one of the acknowledged experts is present. However it is not a treatise on the subject but an information brief concerning Yuma Proving Ground and how Yuma intends to implement a distributed computer network. This paper, originally a briefing prepared for the DoD Research & Engineering Directorate (DDRE), contrasts some of the known current distributed computer systems with the type which Yuma intends to install. We tried to show the practicality, desirability, and feasibility of a ring type distributed network for Yuma without getting too bogged down in technical details, since those briefed were management oriented rather than ADPE technically oriented. Notwithstanding, these DDRE managers are at a policy making level influencing the entire scope of ADPE within the DoD. Prior to this briefing their ADPE orientation was towards large stand alone computers, which is diametrically opposed to the ring concept planned for YPG. This is the briefing paper substantially as it was presented. The purpose was to sell a concept and it succeeded in that objective.

At TECOM we have a diversity of weapon systems to be tested. Foremost among these are the Army's big five developments.

- \*Mechanized Infantry Combat Vehicle (MICV)
- \*Advanced Attack Helicopter (AAH)
- \*Utility Tactical Transport Aircraft System (UTTS)
- \*Surface-to-Air Missile Development (SAM-D)
- \*XMI Tank

Both instrumentation and ADPE must be able to react to the testing requirements of these items as well as all other Army materiel in a timely and economical manner.

At YPG for example, these requirements are prompted by testing application ranging from ground vehicles, such as MICV & the XMI Battle Tank, to Artillery such as the XM204 Howitzer; to Aircraft-Armament; and most recently the Global Positioning System, A Tri-Service Responsibility of the Air Force.

Figure 1 depicts the Global Positioning System (GPS) concept. When complete, GPS will provide time and positioning data for any receiving unit at any location on earth. The initial Yuma testing will consist of: (1) A simulated or inverted range, & (2) Testing with up to nine satellites which will be within the reception area of YPG approximately four hours per day.

During both the Aircraft Armament tests and the Global Positioning System tests, large amounts of data will be collected within short time frames. It is essential to do as much processing in real time as possible, in order to speed the analysis of the data collected and to prepare reports in the shortest possible time frame.

Recognizing that a large increase in ADPE workload would result from these testing requirements, TECOM initiated action to review all ADPE at YPG. As a result several alternatives were identified which were then studied, in full recognition of the following dynamic technological thrusts which impact ADP today:

1. Hardware costs are decreasing by a factor of 100 each ten years, with every indication of continuation for the next ten years.
2. Software costs are decreasing by a factor of 10 each ten years, with every indication of continuation for the next ten years.
3. Data communication costs are decreasing by a factor of 10 each ten years, with reasonable expectations of continuation for the next ten years.
4. Personnel costs are essentially stable, after accounting for inflation.

The consequences are that overall - personnel costs are looming larger relative to all other costs, with hardware diminishing as a cost factor.

But most present-day real time data systems have been implemented using large scale computer systems such as at White Sands Missile Range where large UNIVAC 1108's are used in the real time tracking of missiles. These systems effectively control input, output and processing of all data for real time systems. (Fig 2) As you are aware, the cost of these large scale computers is always high, usually several millions of dollars or more.

On the other hand the ever increasing sophistication and capabilities of minicomputers, are rapidly approaching those of large scale computers. This is accomplished by separating the various processing elements used in a real time system and substituting interconnected minicomputers for each. The resulting multi-processor system, using a number of minicomputers, can do the same real time job as a large scale system and for much less cost.

As a result, the thrust of our current thinking is more in terms of small minicomputers. (Fig 3) For this reason new practical and cost effective solutions for meeting the future ADPE requirements of YPG were considered along with the traditional approach.

With the most recent availability of low-cost minicomputers a computer trend toward localized computing at the site of the user is developing in industry and in Government as well. For example, here in TECOM the computer has become an integral part of instrumentation, test chambers, and data acquisition and control systems. This requires that at least part of the computing facility be at the application site. The minicomputer has been ideally suited to such real-time applications. This leads us to a concept of integrating a number of minicomputers into a computing system. (Fig 4)

In effect this is an information utility made up of a number of minis rather than one or two maxi-computers. This concept is known as a distributed computing system. The goal of distributed computing is an integrated hardware system which provides reliable service at low cost.

There are several different kinds of distributed mini-computer networks in existence today - all of which are in various stages of development. Representative of these are:

- \*The Carnegie-Mellon Multi-Mini Processor
- \*The Bell Labs Spider Network
- \*The University of California at Irvine Ring Network

Figure 5 is a schematic of the ADPE contained in the Carnegie-Mellon Multi-Mini-Processor or C.mmp. This system will contain up to sixteen mini-processors, five of which are shown. These are connected through a switch to memory boxes.

The switch allows any processor to access any of the memory boxes.

The processors are not permanently attached to a memory box, rather each time a processor wishes to access a particular memory a connection is established through the switch for that access, sixteen separate processor-memory connections will be possible simultaneously.

Peripheral devices are connected to buses associated with each processor and gain access through these buses to the shared memory. Each processor can intercept each of the other processors at several priority levels and can start and stop other processors. This enables one processor to task another to perform an operation for a program running on the first processor. Thus user programs are not restricted to execute on any particular processor.

Figure 6 illustrates the techniques used to keep the highest priority jobs in processing. If a new user program enters the queue and is a higher priority than the job in "A" then the priority will be compared with the priority of the job in processor "10". If the new job is a higher priority it will take the place of the job in processor "10" and that job will be returned to the waiting queue as the highest priority in the queue.

An algorithm stored in the memory controls the matching of priority jobs and processors, and insures that those jobs with the highest priority are processed first.

There are several benefits to the C.mmp - (1) by having multiple processor units, the failure of any one will not crash the total system. Removal of one processor from the system will affect the system so little that it may hardly be noticed.

(2) Minicomputers are produced in large quantities at low cost, and as it turns out the C.mmp system costs less than one half of what a single machine of similar power would cost.

(3) Interconnected minis allow user organizations to start at a level using only the number of minis needed and, as requirements and usage grows, expansion can be achieved by adding processors as required. This technique is much more cost effective than replacing a large processor.

(4) In the C.mmp configuration, if required all the processors may cooperate to solve a single problem or each processor may be dedicated to a different user or any combination in between.

The C.mmp is not a geographically distributed network. All processors are in the same room. It is however, an alternative to a large maxi-computer.

Figure 7 shows a Bell Labs type Distributed Network. This network has a central control known as "Spider". All communication must first go to Spider for messages to be routed to the appropriate network member. This creates a



central decision point facilitating workload distribution and resource sharing. Depicted are different configuration sets consisting of three mini's each. (The Bell Lab system actually links eleven mini-computers of five different types.) Each machine connects to Spider through a terminal interface unit (TIU). The basic idea for this system involves the transmission of data between minicomputer terminals in packets, or bursts, rather than a uniform stream. Each mini-computer terminal is associated with a buffer. Each buffer is large enough to hold at least one data packet. This method provides the time periods needed for making effective routine decisions by Spider which operates as a data switch.

Spider is the central component of the system and switches data between the various minicomputers. Each of the minicomputer terminals sends out data bursts in response to encoded control signals generated by a Spider controlled interface technique. Data transmitted from a mini terminal is picked out by a multiplexer at intervals determined by the encoded interface control signals.

In doing this, the multiplexer assembles the data into the proper packet format prior to sending it to Spider on one of the network transmission lines. Packets of data are transmitted by placing messages into an open time slot on a conveyor belt-like channel. Input messages are received similarly.

The advantages of the Spider Network are the same as for the C.mmp. However, this system is distributed over a larger area, being specifically designed to support different laboratories conducting experiments and research.

One potential shortcoming of the system is the switch communication computer, the Spider, thru which all traffic flows. Failure of this component causes a failure in the total network.

Figure 8 depicts the University of California (UCI) type system, which has a number of minis connected to a single transmission line in a ring configuration. Each mini interfaces with the line via a device called a ring interface, or RI. Each of the RI units is programmed to impart outgoing and incoming computer information.

Each RI recognizes information on the line addressed to its associated mini and passes it on to that mini while rejecting information not so addressed. The transmission line is designed so that all information flows in one direction. Since all RI's continually reject or accept information from the line, they can spot available time slots in which outgoing information can be inserted without interfering with other data.

Thus, control of data flow and destination is time distributed around the ring. If one or more of the minis or RI's breaks down, the rest of the system continues to function.

The RI's are equipped with self-monitoring circuits that detect any malfunctioning of normal operations. Since RI or mini failure could cause road blocking of the transmission line, the self-monitoring circuit acts as a circuit breaker to take the RI and its mini off the line if appropriate.

The only store-and forward message switching technique required in the system is that between each mini and its RI. Besides greatly reducing transmission time, this simplifies the RI design, thus the RI's can be built for much less than any other type of communications processor - less than five hundred dollars. This further enhances cost effectiveness when the ring is expanded by hooking up an additional RI to the line and connecting it to a mini.

Each RI contains a multi address code. When a message from another RI passes it on the line, the destination code is compared to the stored addresses. If a match results, the message is accepted and the information transferred to the minicomputer. If the minicomputer cannot accept the message because it is occupied, the message is transmitted to the next appropriate computer in the ring.

Figure 9 depicts the completed UCI ring network including its associated peripherals, (i.e., printers, plotters, & disks). Low cost is achieved here in several ways. Each of the component computers of the network is relatively small and inexpensive. The system software is a modest programming effort, existing other software can be integrated readily, and finally standardized interfaces to the communication ring are available at low cost as previously indicated.

The same benefits apply here as in the previous networks. However it is even more reliable because no central control computer is required as in the Bell Lab's system. Also the ring concept has the advantage of allowing for greater geographical dispersion than the C.mmp.

Having examined 3 examples of minicomputer distributed networks, let us turn now to TECOM.

Profiting from the R&D efforts of others, TECOM at Yuma Proving Ground is moving into this new technology with the expansion of the real-time data acquisition network on the Aircraft Armament Range. This will be the first non-laboratory application of the ring concept. The ADPE (some now in operation) to be included in this distributed network will range from programmable calculators, to minicomputers, to a large-scale central real-time and batch processing system.

Shown schematically in Figure 10 is the existing instrumentation/ADPE for the Aircraft Armament Range for Yuma's Cibola Range.

Two computers (an EMR 6130 and an IBM 7044/7094 Direct Coupled System) comprise the main computer site. A real-time cinethedolite data system (CINE), located at three separate sites tracks a target and transmits position data to the main computer site in real-time. A precision aircraft tracking system(PATS),

which is a Laser Tracker, also tracks and transmits position data to the central site. The data flow illustrated occurs five times/sec. Total computation time for all processors in the central computer takes under one hundred eighty milliseconds.

Figure 11 is a schematic of how the range will look with the ADPE instrumentation system now in procurement. The two additional laser trackers (PATS 2 & 3) each have a minicomputer for control, data logging, and formatting. The position locating system at site 7 (PLS) also contains a minicomputer. All data from these data collection systems will be transmitted to the main computer site for real-time data reduction with output then being sent back to the test site for analysis and control by the test officer. The output displays (DISP) also contain minicomputers.

In addition to the instrumentation and ADPE currently in procurement; the central real-time computers at YPG are also expected to be replaced. The replacement will take the form as shown in Figure 12 and represents the new technology in software and hardware which we referred to earlier as a distributed computer network. The net will incorporate key minicomputers at the test sites for data reduction and processing. In this net execution of programs both real-time and batch, will automatically be performed on the optimum system available.

Benefits which are expected to accrue from the YPG distributed computing network are: a higher level of performance; increased reliability and availability because of the redundant nature of the network; and since the net is fundamentally a complex of minicomputers - expandability. In the case of YPG, the capital investment for the basic mini-ring network is much less than it would be for a new stand alone-system. That alone would make it worthwhile. Additionally, by taking advantage of some minis currently at YPG the capital cost of this network will be even less, while operating expense for this system will be approximately seventy-eight thousand dollars per year less than the current systems.

Since the new equipment will cost three hundred thirty thousand dollars it will pay for itself in a little over four years, primarily from the savings in maintenance cost.

In conclusion, a real challenge we face at TECOM is the increasing sophistication of DoD materiel and the demand this places on testing technology. The YPG distributed computer system is but one example of what we are doing in TECOM to enhance our testing capability in a cost effective manner.

# GPS CONCEPT

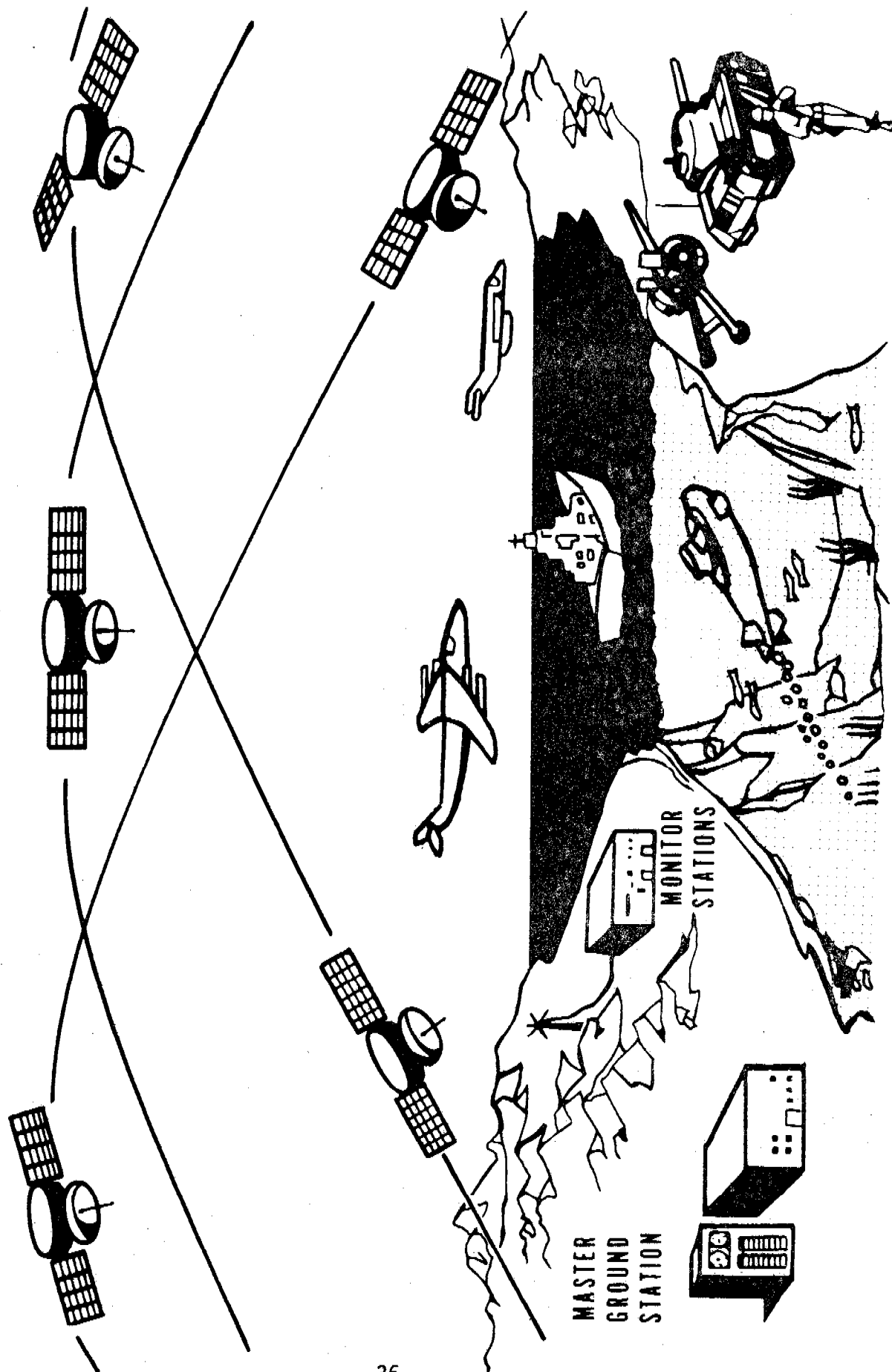


FIGURE 1

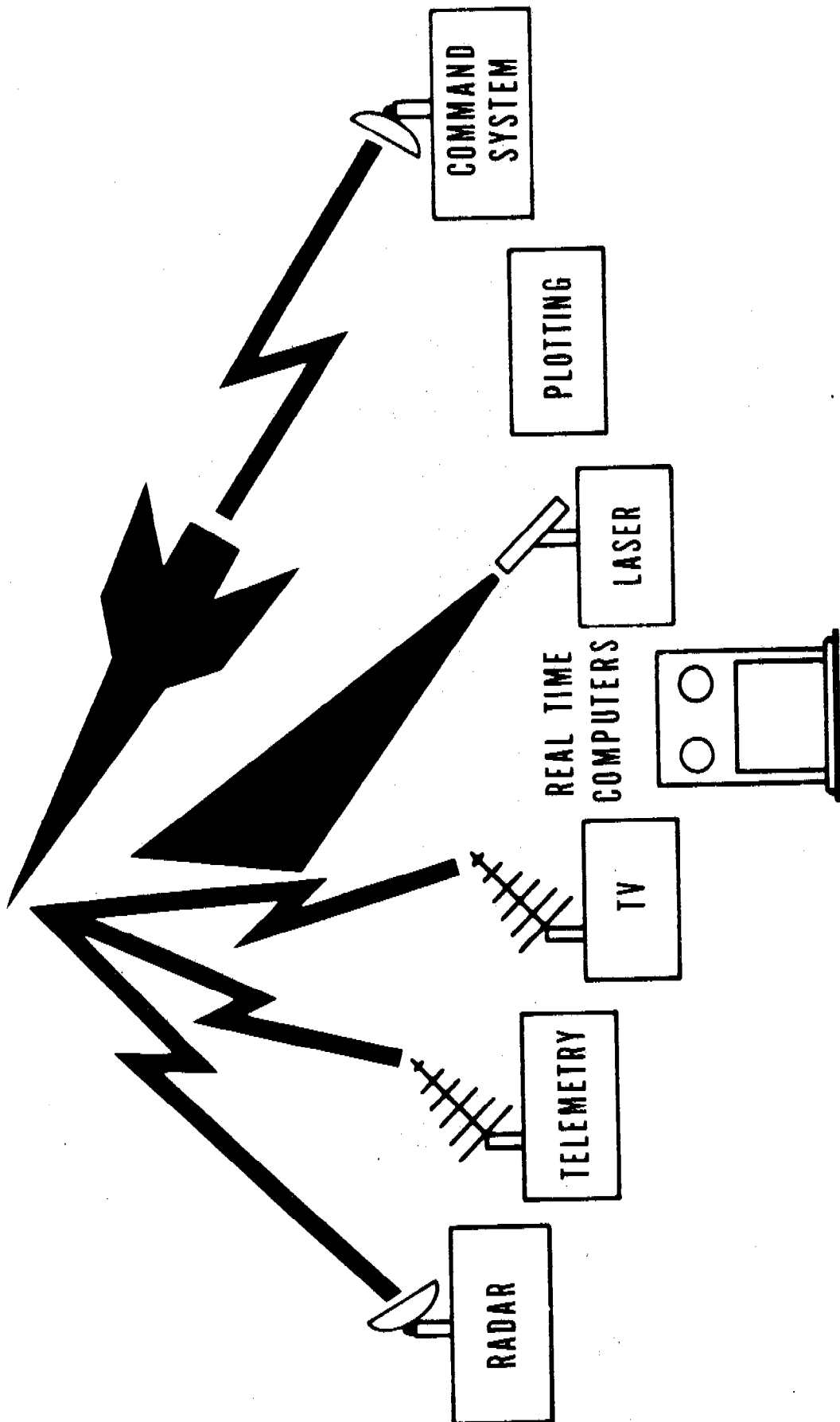


FIGURE 2.

**think**  
**small**

FIGURE 3



**WE CHALLENGE YOU !**

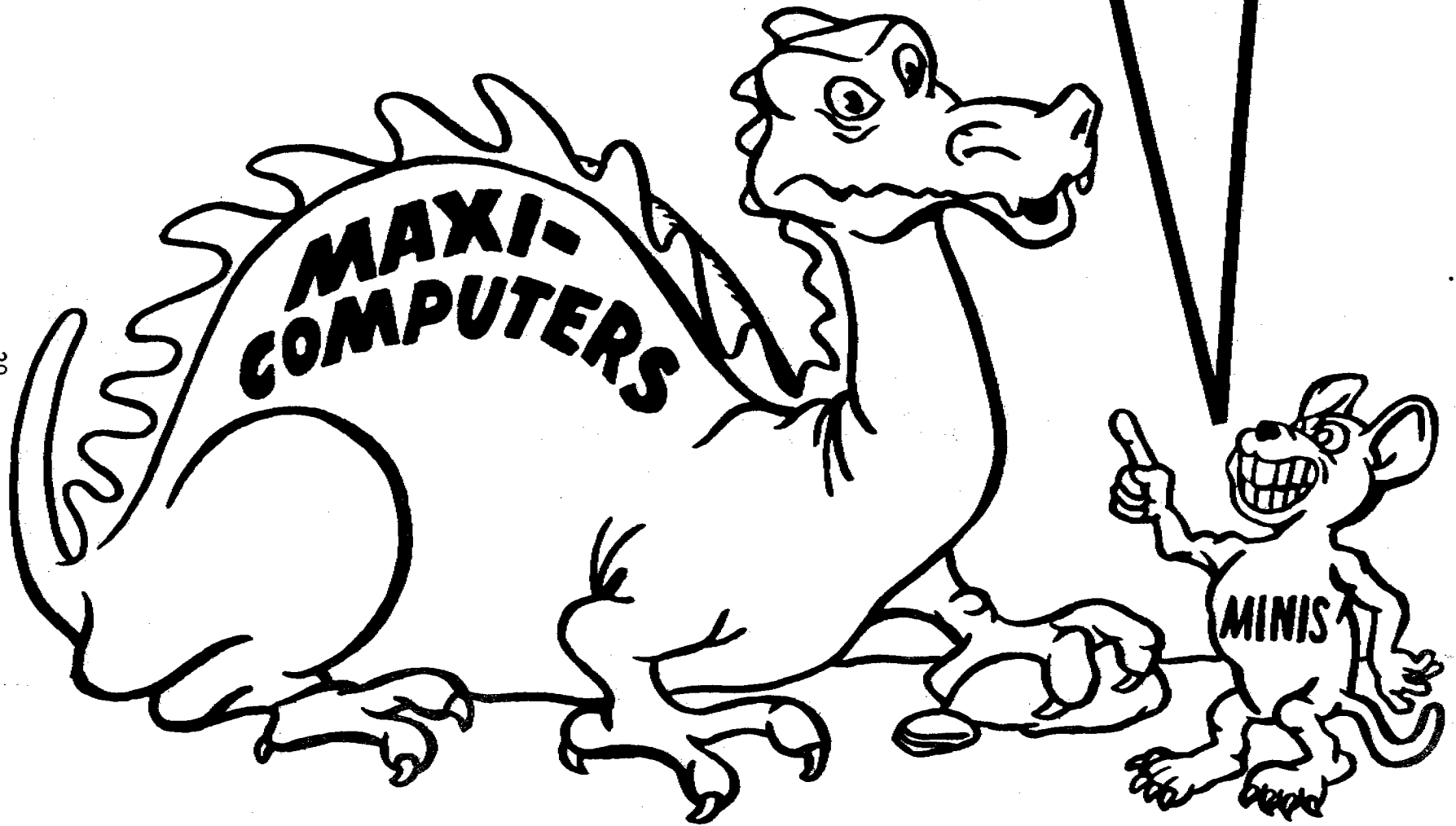


FIGURE 4

# CARNEGIE-MELLON MULTI-MINI PROCESSOR (CMMP)

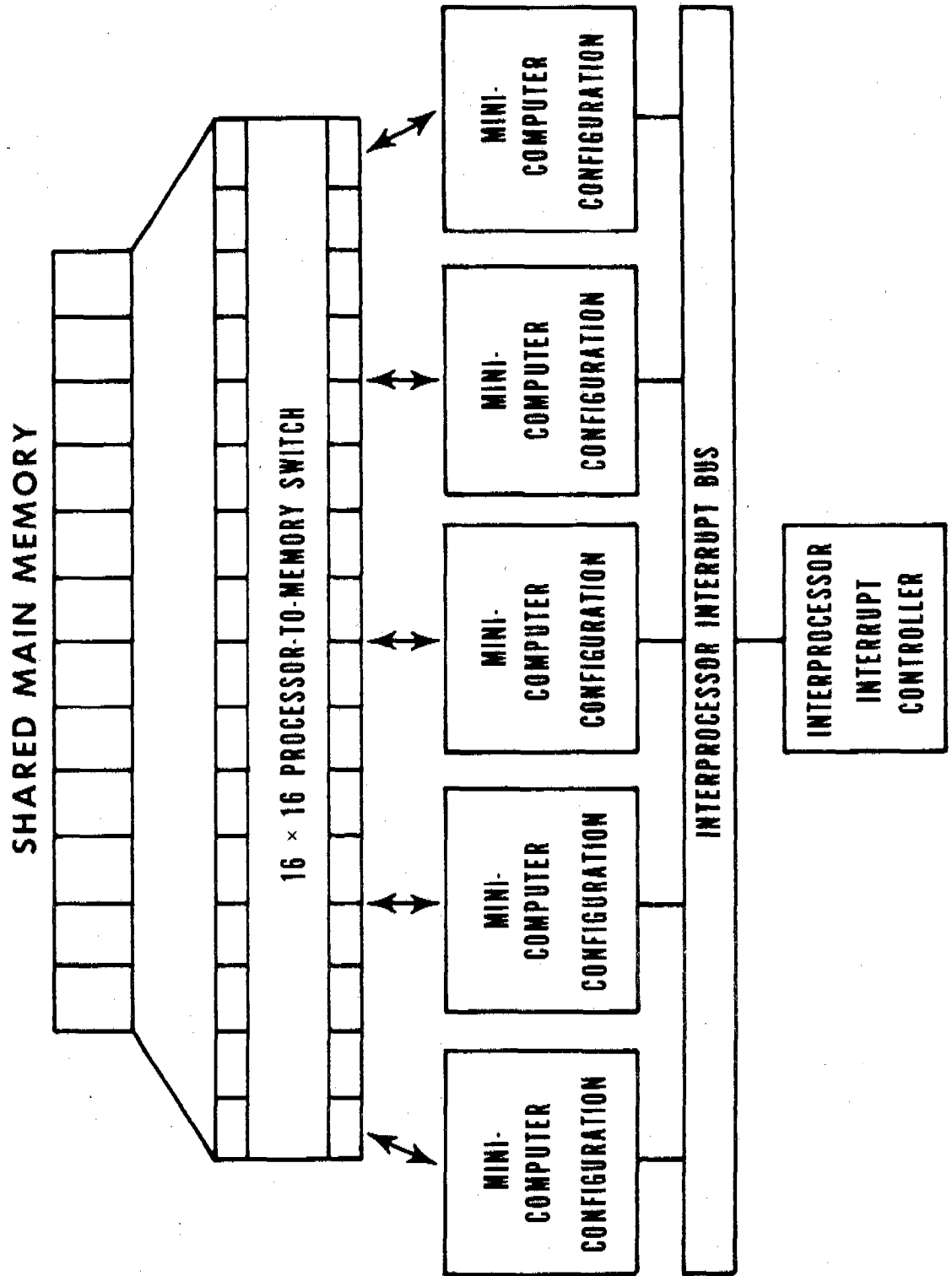
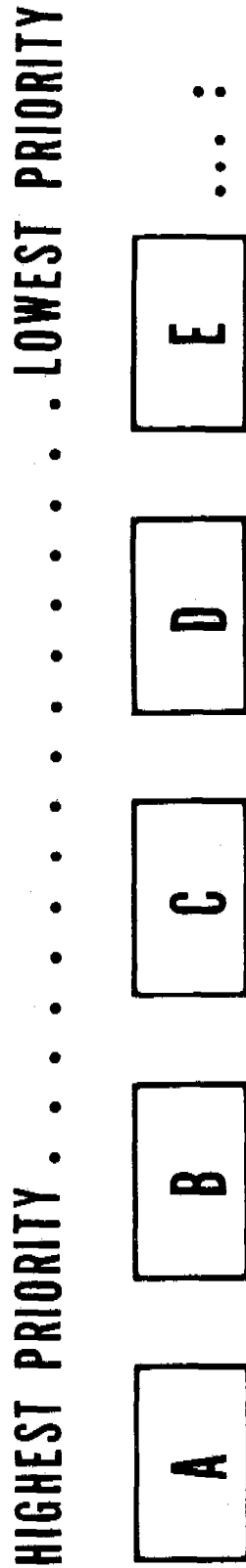


FIGURE 5



# QUEUE



# PROCESSORS

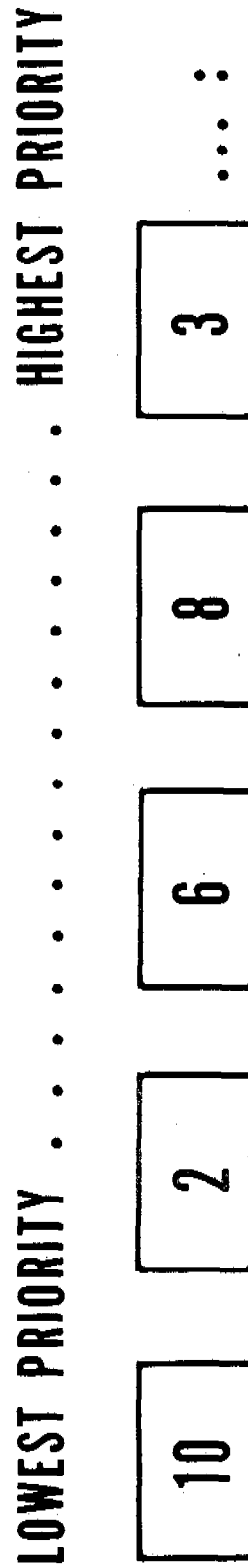
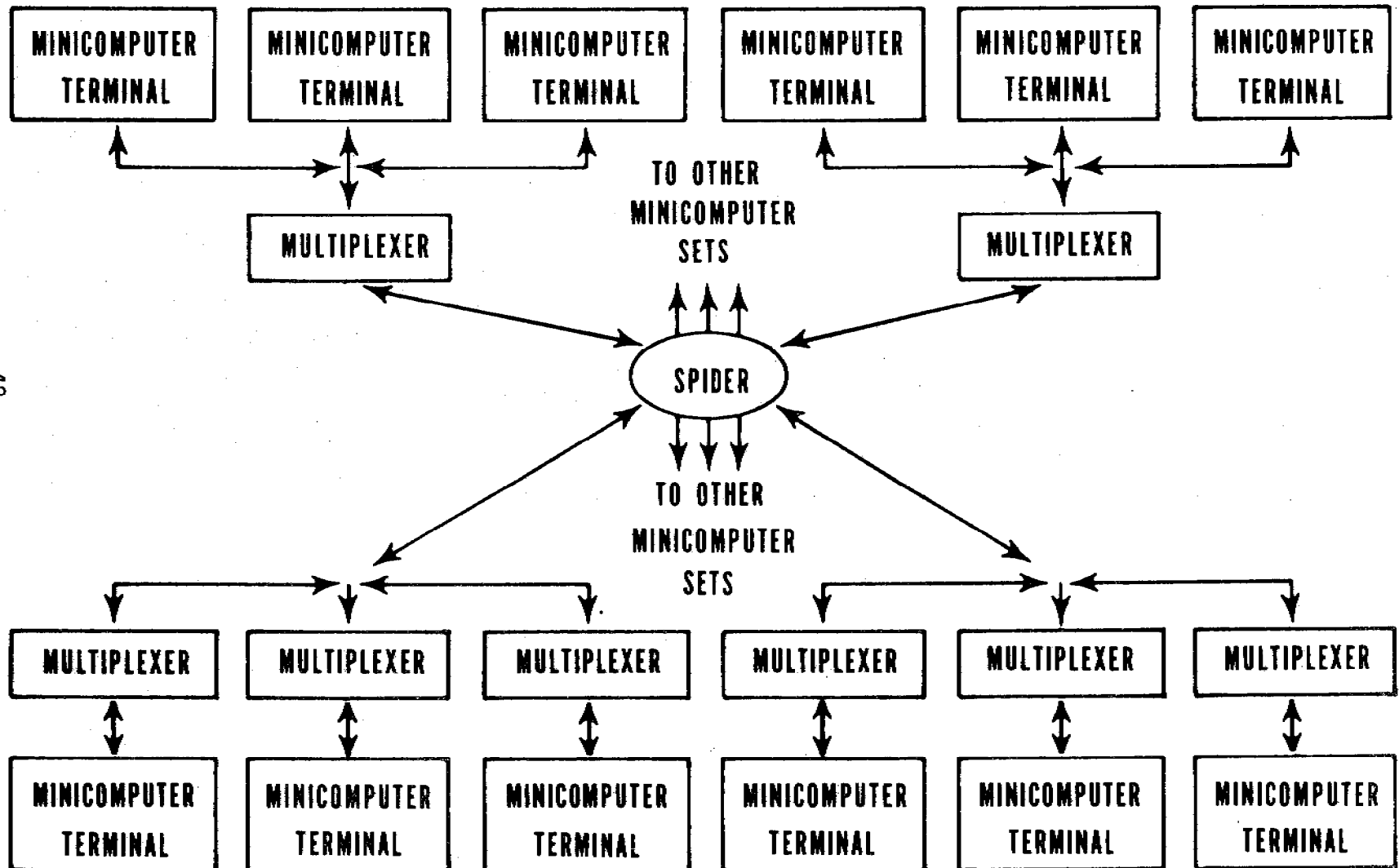


FIGURE 6

# BELL LAB'S SPIDER NETWORK

## MINICOMPUTER SET USING CONCENTRATOR CONFIGURATION



## MINICOMPUTER SET USING LOOP CONFIGURATION

FIGURE 7

# RING NETWORK

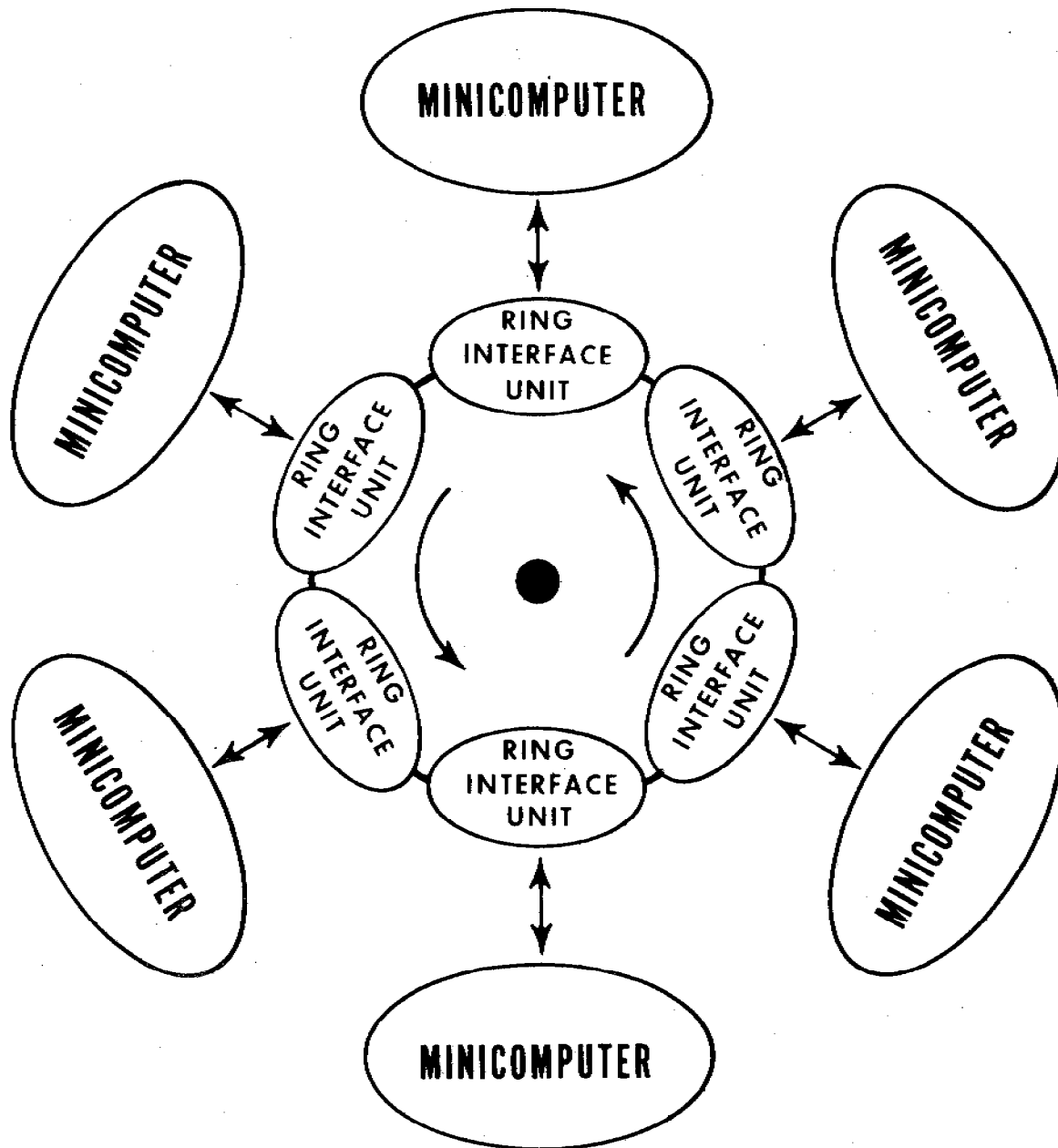


FIGURE 8

# UNIVERSITY OF CALIFORNIA AT IRVINE (UCI)

## DISTRIBUTED COMPUTER SYSTEM RING NETWORK CONFIGURATION

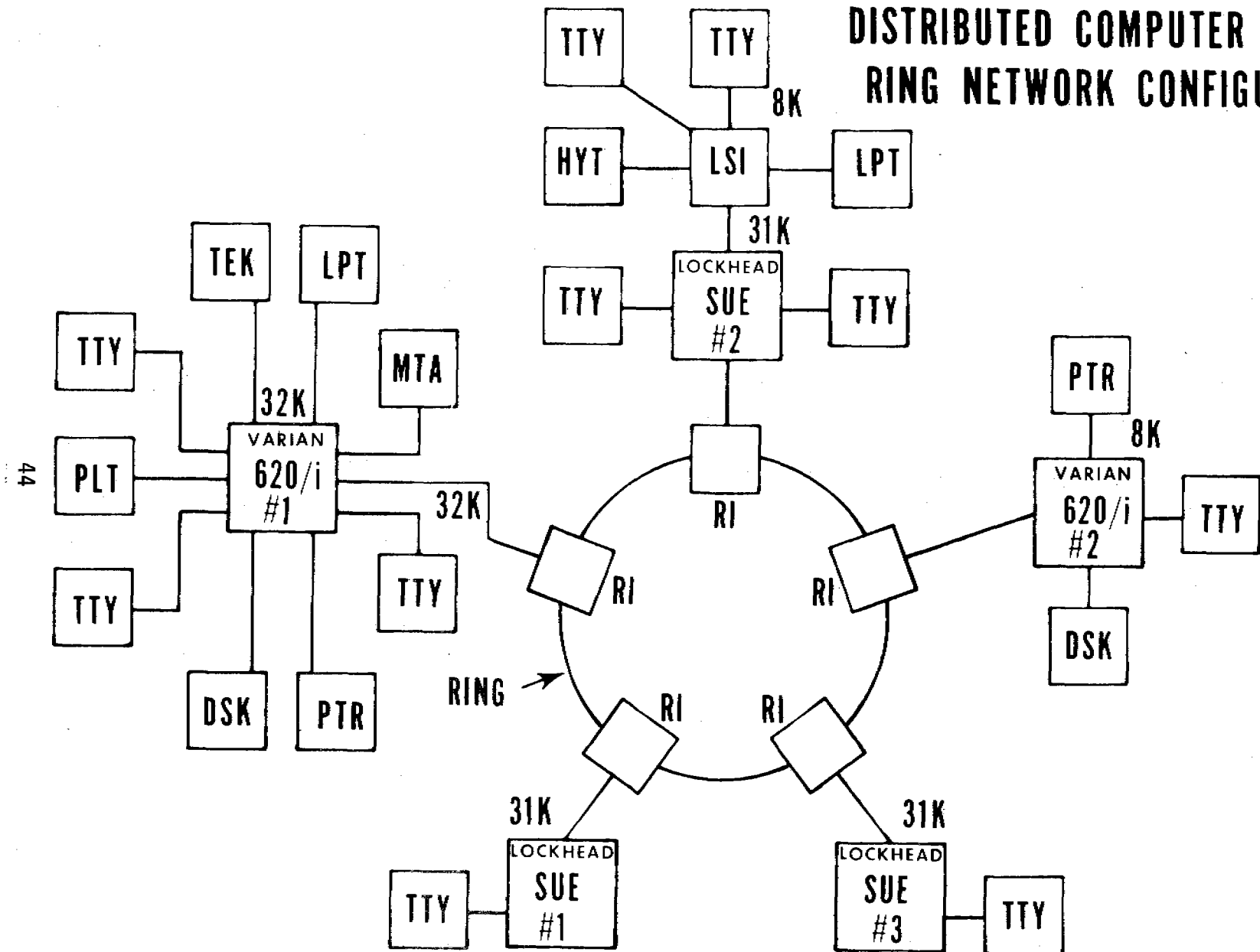


FIGURE 9

# PRESENT YPG AIRCRAFT ARMAMENT RANGE

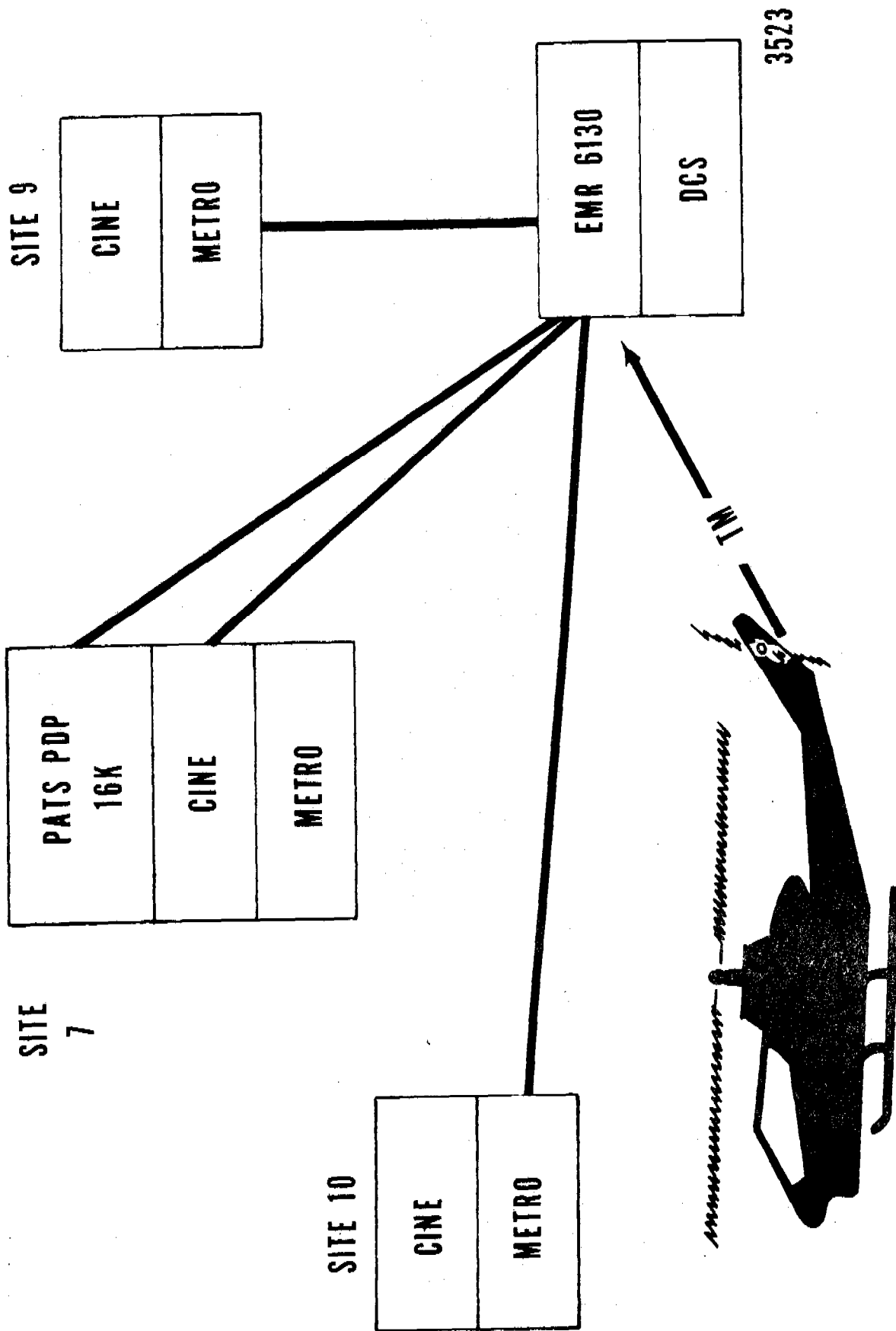


FIGURE 10

YPG AA RANGE (2ND Q FY76)

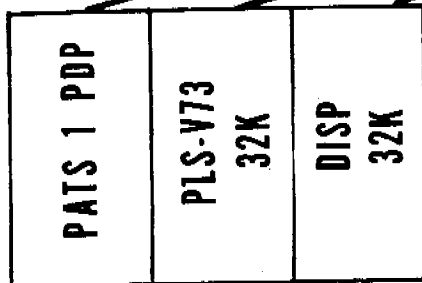
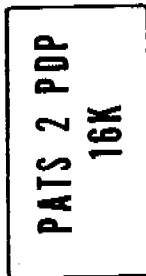
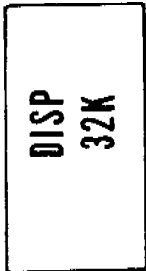
C. D. HELIPORT

SITE 9

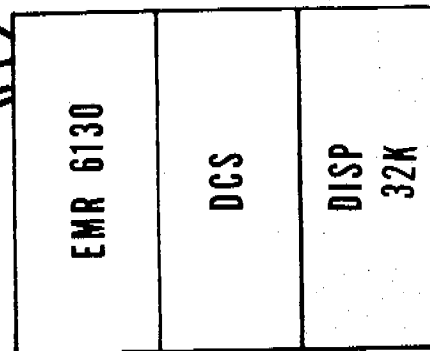
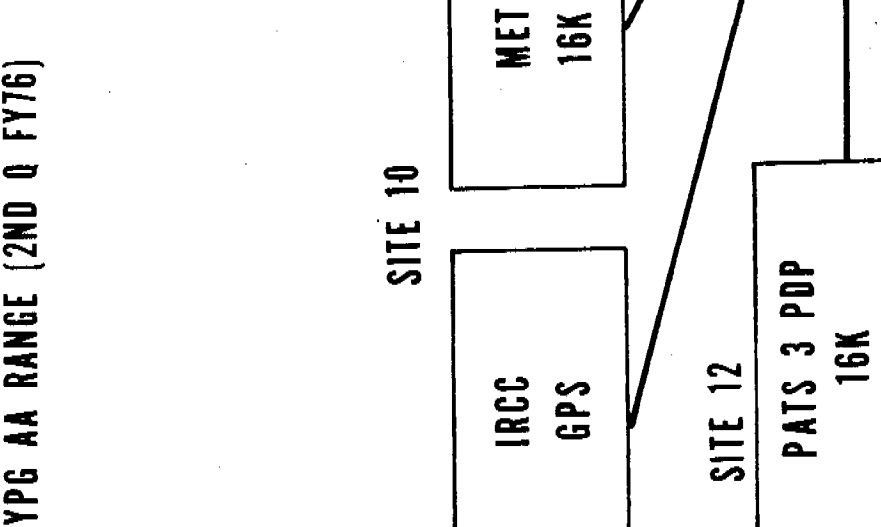
SITE 7

SITE 10

SITE 12



SITE 12



3523



FIGURE 11



# YPG DISTRIBUTED NET.

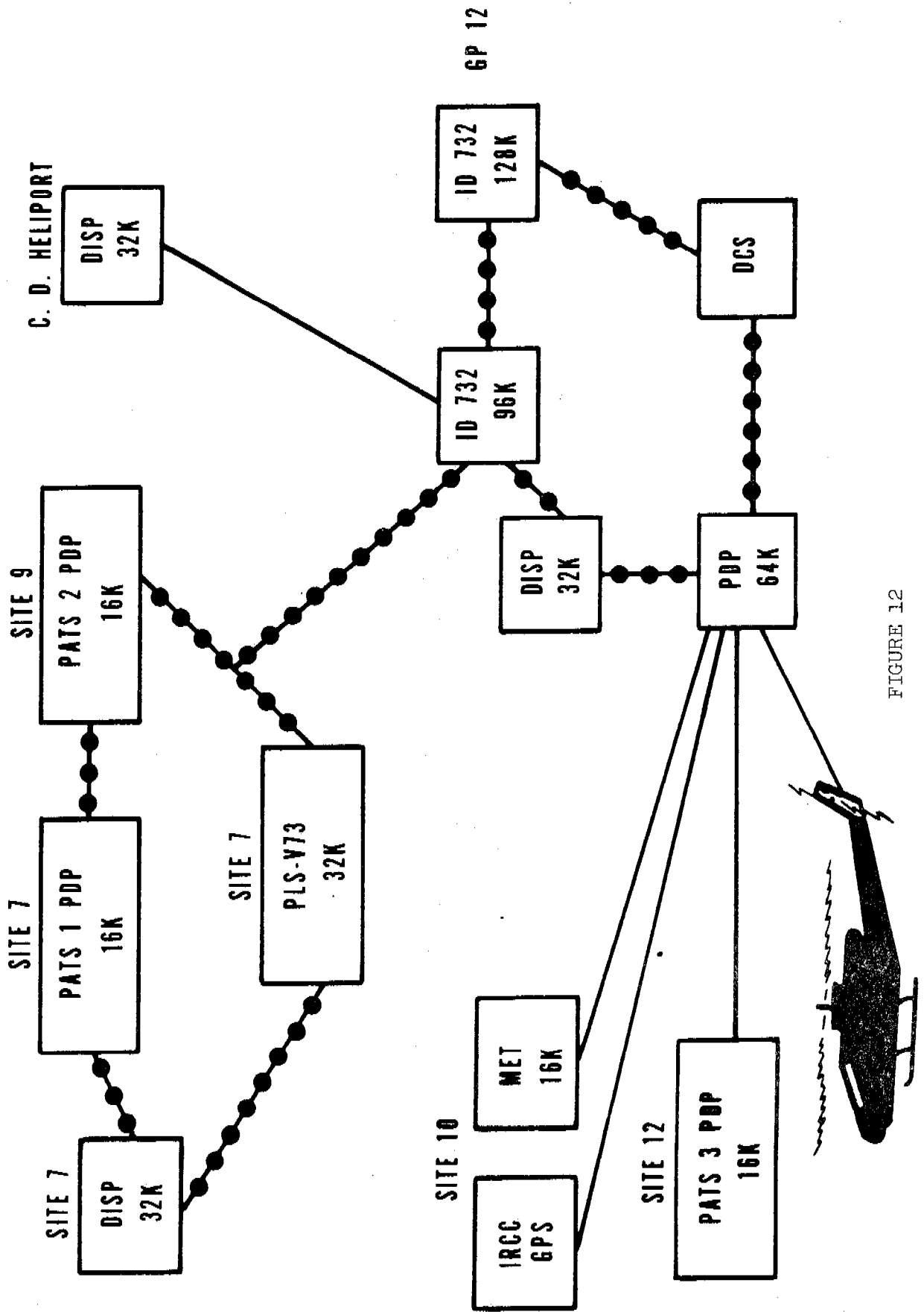


FIGURE 12





# LINEARIZED LEAST SQUARES

Larry M. Sturdivan and John W. Jameson  
Biomedical Laboratory  
Edgewood Arsenal  
Aberdeen Proving Ground, Maryland 21010

**ABSTRACT.** The paper discusses iterative methods for fitting linearizable functions by weighted least squares. Starting values of the coefficients are derived from data. The method is adaptable for some desk top programmable calculators. The effect of various weighting schemes on the residual sum of squares is discussed. Two methods are presented for making the residuals in the dependent variable sum to zero.

**1. INTRODUCTION.** Since the early 19th century the method of least squares has been the most widely used mathematical tool for determining the values of constant parameters in linear functions (linear with respect to the parameters to be determined). The general, nonlinear least squares problem, by contrast, remains a largely unsolved problem. There is, however, a class of nonlinear functions, which some authors call "intrinsically linear", for which good estimates of the least square values of the fitted parameters may be easily and reliably obtained.

**2. STATEMENT OF PROBLEM.** Let us assume that we have a dependent variable  $y$  which may be expressed as an explicit function of a set of independent variables and constant parameters whose values are to be estimated.

$$\text{i.e. } y = F(x_i, b_j) \quad ; i = 1, \dots, n \quad ; j = 1, \dots, m \quad (1)$$

$y$  = dependent variable

$x_i$  = set of  $n$  indep. variables

$b_j$  = set of  $m$  constant parameters

The following algorithm may be used to determine the  $b_j$  if  $F$  may be made linear with respect to the  $b_j$  by suitable mathematical transformation, i.e. if there is some function  $h(y)$  such that

$$h(y) = \sum_{j=1}^m b_j g_j(x_i); i = 1, \dots, n \quad (2)$$

and  $dh(y)/dy$  is defined

The functions  $F$  and  $g_j$  may involve known constants which are not members of  $b_j$ ; e.g. the fourth of the following examples of intrinsically linear functions:

$$y = \ln (bx)$$

$$y = \arctan (bx)$$

$$y = \sum_{i=1}^{n_{\pi}} x_i^{b_i}$$

$$y = \frac{K_0}{K_1 + a \exp (b_i x_i)} \quad K_0, K_1 \text{ Known}$$

Of course conventional least squares could be used directly on equation 2. The result, however, is the minimization of squared errors in  $h$ , not  $y$ .

3. THE ALGORITHM. We designate the difference between the actual and predicted value of  $y$  for the  $k$ th data point as:

$$\Delta y_k = y_k - F(b_j, x_{ik}) \quad k = 1, \dots, p$$

and

$$\Delta h_k = h(y_k) - h(y_k - \Delta y_k) = h(y_k) - h(F(b_j, x_{ik}))$$

Expanding  $h(y - \Delta y)$  in a Taylor series with a remainder term, we get

$$h(y - \Delta y) = h(y) - \Delta y h'(y) + \frac{\Delta y^2}{2!} h''(y) + \dots + \frac{(-\Delta y)^q h^{(q)}(\bar{y})}{q!}$$

where  $h^{(q)}(\bar{y})$  is the  $q$ th derivative of  $h(y)$  with respect to  $y$ , evaluated at  $\bar{y}$ , and where  $\bar{y}$  is in the interval between  $y$  and  $y + \Delta y$ . Let  $q = 1$ , then

$$\frac{\Delta h}{\Delta y} = h'(\bar{y})$$

Now we want to minimize

$$\sum_{k=1}^p (\Delta y_k)^2 = \sum_{k=1}^p \frac{(\Delta h_k)^2}{[h'(\bar{y}_k)]^2} \quad (3)$$

This is equivalent to weighted least squares on equation 2 with weights

$$u_k^2 = [1/h'(\bar{y}_k)]^2; \quad k = 1, \dots, p \quad (4)$$

Initially we do not know the value of the  $\Delta y_k$ , much less the position of  $\bar{y}_k$  in the interval, so we use the data points  $y_k$  to approximate the  $u_k$ . We may then use the  $\Delta y_k$  from the first pass to calculate improved weights and make a second, etc., continuing the iteration until stability is reached. Several methods of obtaining improved estimates of the  $u_k$  of equation 4 have been tried. Some authors have used regression line values

for  $\bar{y}_k$  in equation 4; in effect using the opposite end of the interval from the data point. This method improves the fit (decreases the sum of deviations in equation 3) most of the time, but if  $\bar{y}_k$  is closer to the data point than the regression value the fit thus obtained is actually worse than the results of the first iteration. Other methods tried include:

- a) The first q terms of the Taylor expansion about the data values.
- b) The first q terms of the Taylor expansion about the regression values.
- c)  $u_k^2 = \left(\frac{1}{h'(y_k)}\right) \left(\frac{1}{h'(y_k + \Delta y_k)}\right)$  (geometric mean of data and regression values)
- d)  $u_k^2 = (\Delta y_k / \Delta h_k)^2$  ( $\Delta$ 's from previous iteration)

Methods a and b have the serious drawback of requiring the first q derivatives of h, almost always an unpleasant task whatever the form of h. It is wasted labor in most cases, as well, for method d usually gives results as good as, or better than any other. In fact, intuitively one might think that method d would lead to convergence on "the" least squares answer. However, we must remember that we are determining m weights and n b's from just m data points, an underdetermined system. Thus as long as the weights are determined from the data, as they must be, one cannot guarantee convergence on the global minimum. Indeed, we have observed intermediate iterations in which the sum of squared deviations in y was slightly smaller than the "converged" value.

Unlike the linear least squares case, the above algorithm does not produce a zero sum of deviations in y:

$$\text{i.e. } \Sigma \Delta y = \Sigma u \Delta h \neq 0$$

$$\text{Although, } \Sigma u^2 \Delta h = 0$$

If one has an overriding reason for desiring a zero sum of deviations in y, it may be obtained by minimizing

$$\Sigma u (\Delta h)^2 = \Sigma \Delta y \Delta h \quad (5)$$

which, of course does not minimize the sum of squared deviations in y. In many cases the increase in squared deviations is tolerably small with minimization of equation 5 (i.e. using weights u rather than  $u^2$ ) if zero sum of deviations is desired. A second method of obtaining a zero sum of deviations is to force the fitted line to pass through the weighted mean  $h^*, x_i^*$ . i.e. fit the model

$$h - h^* = \sum_{j=1}^{m-1} b_j g_j (x_i - x_i^*) \quad i = 1, \dots, n$$

where  $h^* = \frac{\sum u h}{\sum u}$

and  $x_i^* = \frac{\sum u x_i}{\sum u}$

using weights  $u^2$ . The summation running to  $m-1$  indicates the absence of an intercept; i.e. one of the  $b$ 's, say  $b_f$ , for which  $g_f = 1$ . Both methods of obtaining  $\sum \Delta y = 0$  work best when the  $u$ 's are calculated by method (d) above.

4. AN APPLICATION. An example of an intrinsically linear function is the Logistic (probability distribution) function

$$y = \frac{1}{1 + a/x^{b_1}}$$

The linearized form is

$$h(y) = \ln(y/1-y) = \ln a + b_1 \ln x = \sum_{i=1}^2 b_i g_i(x)$$

where  $g_1 = \ln x$

$$g_2 = 1$$

$$b_2 = \ln a$$

Also

$$u = \frac{1}{h'(\bar{y})} = \frac{1}{\bar{y}(1-\bar{y})}$$

The following table shows how much improvement in the root mean square error was gained by using the iterative algorithm on the Logistic function over an unweighted fit. As expected, the fit is perfect when the data all lie right on the fitted line, whether the least square fit is weighted or not. As the data get more scattered (i.e. with a correlation coefficient significantly less than 1.00) the reduction in root mean square error is marked (Case III).

Case	Correl. Coeff.	Root Mean Sq. Unweighted	Error Weighted	% Improvement
I	1.00	0	0	0
II	.98	0.0496	0.0488	2
III	.94	0.0419	0.0368	12

# NONLINEAR SPLINE REGRESSION ON MINI-COMPUTERS

Philip W. Smith\*

## Appendix

Stanley Hrnecir and Philip W. Smith\*

Department of Mathematics  
Texas A&M University  
College Station, Texas

1. Introduction. In this paper we will present our experiences in attempting to solve certain nonlinear regression problems on a mini-computer. More specifically, we used spline curves of a specified order and treated the knots as nonlinear parameters.

Section 2 contains the relevant notation and theory. In section 3 we make some final remarks. At the end is an appendix with a listing of the programs and an abbreviated description of their use with examples. The reader who is unfamiliar with splines should first glance through the examples in the appendix.

2. Description of Numerical Solution. Throughout this section and the next the following notation will be used. Let  $k$  be a positive integer and  $\underline{t}$  be a knot vector satisfying

$$\underline{t} := a = t_1 = \dots = t_k < t_{k+1} \leq \dots \leq t_n < t_{n+1} = \dots = t_{n+k} = b$$

where  $t_{j+k} > t_j$ . Given a  $k$  and a knot vector  $\underline{t}$  one forms the normalized B-splines  $N_{j,k}$  by [1],

$$N_{j,k}(\underline{t}, \tau) = (t_{j+k} - t_j)[t_j, \dots, t_{j+k}](\cdot - \tau)_+^{k-1},$$

where  $[t_j, \dots, t_{j+k}]$  denotes the  $k$ -th divided difference operator. A spline of order  $k$  with knot vector  $\underline{t}$  is any function of the form

$$s(\underline{t}, \underline{A}, \tau) = \sum_{j=1}^n A_j N_{j,k}(\underline{t}, \tau).$$

\*This author supported by the U.S. Army Research Office under Grant Number DAHC 04-75-G-0186.

Note:  $(s-v)_+^{k-1} = (s-v)^{k-1}$  if  $s > v$  and is zero otherwise.

Given data  $\{(x_i, y_i)\}_{i=1}^L$  one can attempt to find  $\underline{t}^*$  and  $\underline{A}^*$  so that

$$\min_{\underline{t}, \underline{A}} \sum_{i=1}^L [s(\underline{t}, \underline{A}, x_i) - y_i]^2 = \sum_{i=1}^L [s(\underline{t}^*, \underline{A}^*, x_i) - y_i]^2.$$

It is in general impossible to find  $\underline{t}^*$  and hence  $\underline{A}^*$ ; however, given a knot vector  $\underline{t}$  and linear parameters  $\underline{A}$  one can attempt to decrease the error sum of squares by some sort of descent algorithm.

The method of descent which we use is essentially that of deBoor and Rice [2].

Algorithm: Given  $\underline{t}$  and  $j = k + 1$

Step 1: Find  $\underline{A}$  which minimizes

$$\sum_{i=1}^L [s(\underline{t}, \underline{A}, x_i) - y_i]^2.$$

Step 2: With  $\underline{A}$  fixed, move  $t_j$  between  $t_{j-1}$  and  $t_{j+1}$  so as to decrease the error sum of squares. Call this new knot sequence  $\underline{t}$ .

Step 3: If  $j < n$  set  $j = j + 1$ , otherwise set  $j = k + 1$ .

Step 4: Go to 1.

This algorithm is described in more detail in the appendix. We remark that Step 1 consists of solving a linear system and that in Step 2 we use a quadratic interpolation to obtain an approximation to the minimum.

We originally had a Fortran version of the above algorithm and "transcribed" this program into Basic when implementing it on an HP 9830A. Some of the characteristics of the HP 9830A are detailed in the appendix. Operating time on the HP 9830A was quite long. A typical problem with 20 data points and  $k = 4$  with 2 interior knots might easily take thirty minutes.

We discovered that most of the time was being spent in evaluating the error sum of squares,

$$\sum_{i=1}^L [s(\underline{t}, \underline{A}, x_i) - y_i]^2.$$

This term must be evaluated quite often during Step 2 of the algorithm. Additional time was being spent filling out the matrix to solve for the linear parameters. The two routines which were being used to evaluate the spline or fill out the matrix had the following form:

Given  $t_i \leq T < t_{i+1}$ :

```
620 N[1,1] = 1
630 FOR S = 1 TO K - 1
640 P[S] = T[I+S] - T
650 M[S] = T - T[I+1-S]
660 N[1,S+1] = 0
670 FOR R = 1 TO S
680 Z9 = N[R,S]/(P[R] + M[S+1-R])
690 N[R,S+1] = N[R,S+1] + P[R] * Z9
700 N[R+1,S+1] = M[S+1-R] * Z9
710 NEXT R
720 NEXT S
730 RETURN
```

This computes simultaneously the values of all the non-zero normalized B-splines at the point T, for the matrix computations.

```
880 FOR R = 1 TO K
890 P[R] = T[I+R] - T
900 M[R] = T - T[I-K+R]
910 D[R,1] = A[I-K+R]
920 NEXT R
930 FOR S = 1 TO K - 1
940 FOR R = S + 1 TO K
950 D[R,S+1] = (M[R] * D[R,S] + P[R-S] * D[R-1,S])/(M[R] + P[R-S])
960 NEXT R
970 NEXT S
980 RETURN
```

This computes  $s(\underline{t}, \underline{A}, T)$  for the error sum of squares.

The parameter K is set at the first of the program and determines the order of the spline. Since K does not change throughout the program we decided to rewrite these routines in such a way that no loops were

involved. This simple expediant reduced the running time by a factor of two or threc. We list below the corresponding changes in the  $k = 2$  (linear spline) program and the  $k = 4$  (cubic spline) program. Given  $t_I < T < t_{I+1}$ ,

For  $k = 2$

```
1892 P1 = T[I+1] - T
1894 M1 = T - T[I]
1896 Z9 = 1/(P1 + M1)
1897 S1 = P1 * Z9
1898 S2 = M1 * Z9
```

This computes simultaneously the values of all the nonzero normalized B-splines at the point  $T$  and stores them in  $S1$  and  $S2$ .

```
2788 P1 = T[I+1] - T
2792 M2 = T - T[I]
2802 C1 = (M2 * A[I] + P1 * A[I-1])/(M2 + P1)
```

This computes  $s(\underline{t}, \underline{A}, T)$  for the error sum of squares. For  $k = 4$  we have

```
1730 P1 = T[I+1] - T
1740 P2 = T[I+2] - T
1745 P3 = T[I+3] - T
1750 M1 = T - T[I]
1760 M2 = T - T[I-2]
1770 Z9 = 1/(P1+M1)
1780 S1 = P1 * Z9
1790 S2 = M1 * Z9
1810 Z9 = S1/(P1 + M2)
1820 S1 = P1 * Z9
1830 C2 = M2 * Z9
1840 Z9 = S2/(P2 + M1)
1850 S2 = C2 + P2 * Z9
1860 S3 = M1 * Z9
1861 Z9 = S1/(P1 + M3)
```



```

1862 S1 = P1 * Z9
1863 C2 = M3 * Z9
1864 Z9 = S2/(P2 + M2)
1865 S2 = C2 + P2 * Z9
1866 C3 = M2 * Z9
1867 Z9 = S3/(P3 + M1)
1868 S3 = C3 + P3 * Z9
1869 S4 = M1 * Z9

```

This computes simultaneously the values of all the nonzero normalized B-splines at the point  $T$  and stores them in  $S1$  through  $S4$ .

```

2490 P1 = T[I+1] - T
2500 P2 = T[I+2] - T
2505 P3 = T[I+3] - T
2510 M2 = T - T[I-2]
2520 M3 = T - T[I-1]
2530 M4 = T - T[I]
2540 C1 = (M2 * A[I-2] + P1 * A[I-3])/(M2 + P1)
2550 C2 = (M3 * A[I-1] + P2 * A[I-2])/(M3 + P2)
2560 C3 = (M4 * A[I] + P3 * A[I-1])/(M4 + P3)
2562 C1 = (M3 * C2 + P1 * C1)/(M3 + P1)
2564 C2 = (M4 * C3 + P2 * C2)/(M4 + P2)
2566 C1 = (M4 * C2 + P1 * C1)/(M4 + P1)
2570 RETURN

```

This computes  $s(\underline{t}, \underline{A}, T)$  for the error sum of squares.

It is easily seen that, as  $K$  increases the length of these new routines increases by a factor of  $K^2$ .

3. CONCLUDING REMARKS. When writing programs on mini-computers one has to be extremely careful not to write the programs in full generality, at least for this generation of mini-computers, due to their slow execution time. For our particular problem we found it necessary to write a special program for each order. Initially this takes more time, but in the long run much time is saved.

Concerning our specific problem we found that most of the computer time was spent in the spline evaluation routine. This occurred quite often during the computation of the error sum of squares. Now, originally the spline evaluation was accomplished in two nested do-loops where the outer parameter was  $K - 1$  where  $K$  is the order of the spline. Since in our applications  $K$  was fixed throughout the entire program, it made sense to rewrite the program for a fixed  $K$ , thus eliminating the loops and linearizing the execution. The details are in Section 2.

#### REFERENCES

1. C. deBoor, On Calculating with B-splines, J. of Approximation Theory 6(1972), 50-62.
2. C. deBoor and J. R. Rice, Least Squares Cubic Spline Approximation II--Variable Knots, CSD TR 21, Purdue University, 1968.
3. C. deBoor, "Package for Calculating with B-splines", The University of Wisconsin-Madison, Mathematics Research Center, Technical Summary Report 1333, October 1973.

## APPENDIX

Stanley Hrnecir and Philip W. Smith

### USER'S GUIDE

#### TABLE OF CONTENTS

- I. Man/Machine Interface
  - A. BASIC Computer Language on HP 9830.
  - B. Cassette Storage Capability.
- II. Main Program
  - A. Input of Data and Knot Sequence.
  - B. Calculation of Initial Error.
  - C. Execution of Knot Moving Routine.
  - D. Program Limitations.
- III. Subroutine Descriptions
  - A. EQUATE.
  - B. Plotting Routines.
    - 1. Spline Graph
    - 2. Data Point Plot
    - 3. Error Plot
  - C. Knot Moving Routine
  - D. Derivative Routine
- IV. Examples of Output from Program
- V. Program Listings
  - A. Linear
  - B. Quadratic
  - C. Cubic

## I. Man/Machine Interface

A. The program presented in this paper was written in the BASIC computer language for the Hewlett Packard Model 9830A digital computer. The BASIC language is very similar to FORTRAN.

1. In using the HP9830, the user is provided with a man/machine interface capability. With this interface capability, the program pauses at various points in its execution to allow the user to exercise different options such as plotting the spline, plotting the data points, calculating the derivative, etc.

### B. Cassette Storage Capability.

1. The HP9830A has secondary storage capability in the form of cassette magnetic tapes. Programs as well as data points can be maintained on these tapes. Having this peripheral capability is a tremendous asset especially when dealing with large programs and/or large volumes of data.

Currently, there are three separate versions of the Spline program being maintained on three separate files. The user can select from any of these depending on which degree Spline program he wishes to run (Linear, Quadratic, or Cubic).

## II. Main Program. The main routine is used only for initialization of input data and knot placement.

A. Input Data. The user is provided the option of inputting data through the keyboard or having the program read previously stored data from a cassette tape. With either option, the user must provide the program with two file numbers. If the data is being input through the keyboard, the program will build two new files on the cassette tape.

1. After the program is loaded and running, the program will request the user to input the number of data points. The request will be displayed on the display panel and the user simply types in the number and then hits the execute switch.

2. The next program request is for the means by which the data is to be input--either keyboard or cassette tape. The program will display the message: "WANT TO LOAD DATA FROM TAPE?". The user's response is either "Y" or "N" for yes or no. If no, the program will expect the data to be loaded through the keyboard, and if yes, from files on the cassette tape.

3. With either option in 2 above, the program will next request two tape file numbers--where the data is to be found or where to store the data. The first number gives the file location where the variable X is (will be) stored and the second number gives the file location where the dependent variable Y is (will be) stored.

4. The next program request will be displayed as: "NUMBER OF EQUATIONS?". The value of this variable (called N) depends on the number of internal knots the user proposes to use in his run plus the order of the Spline (degree + 1) he is using. For example, if the user is running the Cubic Spline Program with 10 internal knots, then the value of N would be 14.

5. The next program request is for input of the knots. All knots are input in order from 1 through  $N + K$ , where  $K$  is the Spline order. The program requires that  $K$  knots be stacked at each end of the interval over which the Spline is being fit.
- B. As soon as the knots have all been input, the program immediately calls subroutine EQUATE to find the unique Spline function that minimizes the error or difference between the input data and the Spline curve calculated for the knot sequence as initialized. The program next calls a subroutine to calculate the error and this error value is printed when returned to the main routine. At this point, the program pauses for 3 seconds to allow the user to halt the program if he wishes to branch to some special subroutine. If the user fails to halt the program at this point, control is given to the knot moving routine which starts the optimization process of moving knots to reduce the error.
- C. The program will remain in the knot moving routine until the user halts the program. The knot moving routine will pause periodically to allow the user to branch to another subroutine if he wishes. This pause will occur at the end of each cycle through the knot moving routine, i.e., after all interior knots have been considered as candidates for moving.
- D. Program Limitations. The program is currently set up to handle a maximum of 50 data points and a max value of 30 for  $N$  (number of equations as described in II.A.4.). This restriction is a program limitation where array lengths are currently set at 50 and 30 respectively. By increasing the dimension of certain arrays, the user can increase the max number of data points and value of  $N$ --machine limit for dimension size is 256. However, the program run time on this machine, for large values of  $N$  and large data sets is tremendous. The program assumes the values stored in the independent variable array  $X$  and the knot sequence  $T$  are in increasing order.

### III. Subroutine Descriptions.

A. EQUATE. For all three versions of the Spline program, the subroutine EQUATE can be found starting at instruction number 550 in the program listings. The structure for the routine was taken from the technical report by C. deBoor, [3].

1. In order to decrease the program run time, a number of changes were made to the subroutine as presented by deBoor (page 38 of [3]). The major changes include the following:

- a. The number of DO loops was reduced.
- b. Cholesky's decomposition method was added to solve the system of equations.
- c. Linear arrays are used to store the diagonal elements of the band matrix instead of two dimensional arrays as done by deBoor.
- d. Since the matrix being generated is symmetric, only the main and upper diagonals are generated and stored.

The diagram illustrates a linear chain of nodes. On the left, there is a vertical sequence of nodes labeled  $C_1, P_1, M_1$ , followed by  $P_1, C_2, P_2, M_2$ , and then  $M_1, P_2, C_3, P_3, M_3$ . These nodes are connected by diagonal lines. A central column of nodes is enclosed in a large oval, containing  $X_1, X_2, X_3$ , followed by three dots, and then  $X_m$ . To the right of this oval, there is a vertical sequence of nodes labeled  $B_1, B_2, B_3$ , followed by three dots, and then  $B_m$ . The nodes in the central column are connected to the nodes in the rightmost column by horizontal lines.

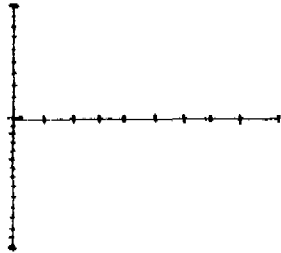
Since the matrix  $A$  is real, symmetric and positive definite, then it is possible to factor  $A$  as  $LL^T$  where  $L$  is a real lower-triangular matrix. Then  $Ax = B$  becomes  $LL^Tx = B$ . Letting  $y = L^Tx$ , then we can solve for  $y$  in  $Ly = B$  by forward substitution and finally we can solve for  $x$  in  $L^Tx = y$  by back substitution. This scheme is known as Cholesky's Decomposition method and is programmed directly into the EQUATE subroutine.

1. Spline Graph. This routine provides the user with a graph of the spline function based on the spline coefficients that are currently in memory. The user will be asked to specify the scaling to use for the graph and step size (distance between calculated points--the smaller the step size the smoother the graph).

a. The first program request reads: "4 SCALE VALUES?". The user inputs 4 values: First two are the limits on the x-axis of the graph and the second two are the limits on the y-axis of the graph. For example, if we know that our input data for the x-axis spans values from 0 to 100 and we expect our spline function to not vary outside the limits of -10 to +10, we might input the following values: 0, 100, -10, 10.

b. The second program request reads: "2 VALUES X-CROSS AND XTIC?". The user inputs the point he wants the x-axis to cross the y-axis and the spacing for the Tic marks on the x-axis. An input, for the example given above, might be: 0, 10.

c. The next program request reads: "2 VALUES FOR Y-CROSS AND YTIC?". The user inputs the point he wants the y-axis to cross the x-axis and the spacing of the Tic marks on the y-axis. For the example given above, an input might be: 0, 1. With the example inputs given in a, b, and c above, the program would sketch the x and y axis as follows:



d. After having the axis plotted, the program will next ask for the step size (Quadratic and Cubic version only). The display simply reads: "STEP = ?". If the user requires a smooth curve, the step size should be small compared to the overall length of the curve. Usually 100 to 200 points provides a very smooth graph. For the example given above, a step size of one would provide 100 graph points of the spline function connected by straight lines which should provide the appearance of a fairly smooth curve. For the linear spline program, a step size is not required because the program simply joins the knots with straight lines.

e. After the spline graph is completed, the program will next automatically sketch in the locations of all interior knots.

2. Data Point Plot. To plot the data points is an option that is usually exercised after the Spline Graph has been completed. This plot provides an "overlay" of where the input data lie with respect to the Spline graph. The routine simply steps through each data point and plots a small "x" at the coordinates of the point.

3. Error Plot. This routine provides a graphical representation of the error at each data point. The error is the difference between the input data point value and Spline Function value for the same point. The routine does its own scaling based on input data and on the square root of the sum of squares of the error taken over all data points. Tic marks are provided for each axis and the value of each is printed on the printer.

C. Knot Moving Routine. Once the Main routine transfers control to the knot moving routine, this routine will stay in an endless loop constantly trying to reduce the error. The only way the program will stop is by operator intervention. The program allows such intervention at the end of each cycle through the knot sequence, i.e., after all knots have been considered as candidates for moving. At the end of each cycle, the program pauses for three seconds displaying the message "PLOT". If the user fails to stop the program at this point, the routine branches to the beginning and starts the cycle all over again in moving knots.

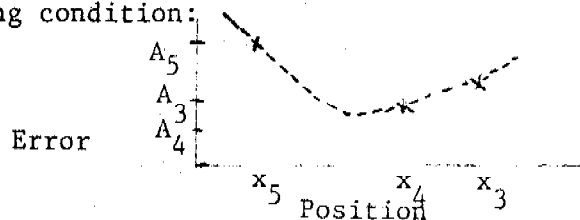
1. Logic of Knot Moving Routine. This routine considers each knot, in order, as a possible candidate for moving. The routine will not change the position of a knot unless the new position decreases the overall sum of squares of the errors.

a. The routine starts at the current position of the knot and steps in the direction of decreasing error. The step size  $E1$  is determined by the variable  $H8$  which is set to 128 by default (user can adjust this value if he wishes--set in instruction 120 of all versions of the program).  $E1$  is initially set to  $1/H8$  of the distance between candidate knot  $X5$  and adjacent knot. Call the interval  $Z$ .

b. The candidate knot's new position,  $X4$ , is determined by  $X4 = X5 + E1$ .  $E1$  can be positive or negative depending on which interval--left or right of current knot--is considered.

c. If position  $X4$  yields an improved error value over  $X5$ , then  $E1$  is doubled and a third candidate position  $X3$  is examined ( $X3 = X4 + E1$ ).

d. If position  $X3$  gives a smaller error value, we then continue stepping through the interval  $Z$  toward the adjacent knot until we either step outside  $Z$  or until a candidate position  $X3$  produces an error greater than the error at position  $X4$ . If this happens, graphically, we have the following condition:



We know that we are in the vicinity of a minimum. We can approximate the position of that minimum by simply fitting a quadratic through the points  $(X5, A5)$ ,  $(X4, A4)$ ,  $(X3, A3)$  and solving for the minimum.

e. After having found a new location for a knot, we call EQUATE, calculate the new sum of squares for the errors, and loop on next knot.

D. Derivative Routine. This routine calculates and prints the zeroth through the second derivative of the Spline Function for 60 equally spaced points along the x-axis for the entire span of data. The value 60 is set by default in the program and the user may change the value if he wishes (instruction 9050 of all versions of the program). This routine is based on the formulae in [1].



IV. Examples of Output from the Program. Output from two separate runs are presented.

A. The first run uses the Quadratic version of the Spline program. Twenty data points are used and two internal knots. The spline was graphed before any knots were moved and again after three cycles through the knot moving routine. The error plot routine was also run after the knots were moved. Figure 1 shows output to the printer.

1. When the Quadratic version of the program was executed, the first two messages printed were:

"QUADRATIC SPLINE PROGRAM"

"K EQUALS 3"

2. The program next requested the number of data points. The number 20 was input and the program printed:

"L9 EQUALS 20".

(L9 is a program variable used for storing data point count).

3. The data was loaded via cassette tape and printed.

4. The next request was for the number of equations; the number 5 was input and "N = 5" was printed.

5. The program next requested the input of knots and each was printed by the program when input.

6. After all knots were input, the program calculated the initial sum of squares of the error and printed the message: "SUM OF SQUARES = 10.24179658". At this point, the program was halted and control transferred to the Spline Graph Routine. The first graph (labeled Plot 1 in fig. 2.a) was accomplished. The Data Point Plot was also accomplished at this point.

7. Control was then given to the knot moving routine. As each knot was moved, the new location and the new sum of squares of the error were printed. As can be seen in figure 1, the two internal knots were moved from 2.1 and 7.0, as input, to 3.213337131 and 6.893706417 with a 10 fold reduction in the error. Three cycles through the knot moving routine were required to accomplish this reduction.

8. The program was halted at this point and control transferred to the Spline Graph routine again. The new Spline Graph (Plot 2 of fig. 2.a) was accomplished and the results show a much better fit to the input data.

9. After the Spline Graph was completed, an error plot was done (fig. 2.b). The scaling was done automatically and the message:

"XTIC = 0.855 YTIC = 0.458175936" provides the user

with the value of the X and Y tic marks.

B. The second example used the same data set but was run using the Linear Spline program and six internal knots.

1. The output from the printer is shown in figure 3 and the plotted output in figure 4.

2. The program was again allowed to cycle three times through the knot moving sequence. The plots were made after the third cycle.

C. Figure 5 provides an example of output for the derivative routine. The routine calculates and prints the zeroth, first, and second derivative for twenty equally spaced points along the Quadratic Spline curve generated for example 1 above.

Figure 1: PRINTER OUTPUT FOR EXAMPLE 1

QUADRATIC SPLINE PROGRAM  
K EQUALS 3  
L9 EQUALS 20

1	0.45	0.532832779
2	0.9	1.135824019
3	1.35	1.634336624
4	1.8	1.850310499
5	2.25	1.653405543
6	2.7	1.004342719
7	3.15	-0.021648662
8	3.6	-1.239057241
9	4.05	-2.385288895
10	4.5	-3.176972882
11	4.95	-3.377363166
12	5.4	-2.859228 04
13	5.85	-1.64757377
14	6.3	0.069777687
15	6.75	1.968942823
16	7.2	3.650872173
17	7.65	4.724967051
18	8.1	4.897943545
19	8.55	4.04801103
20	9	2.266651669

N= 5  
T(I)= 0 1  
T(I)= 0 2  
T(I)= 0 3  
T(I)= 2.1 4  
T(I)= 7 5  
T(I)= 9.3 6  
T(I)= 9.3 7  
T(I)= 9.3 8

SUM OF SQUARES = 10.24179658

2.705905180	3.776486574
7.023652237	3.765599501

3.026696625	1.836770835
6.964153151	1.763237007

3.213337131	1.055100397
6.893706417	0.951943319

XTIC = 0.855      YTIC = 0.109083874

Figure 2a: Example 1 SPLINE PLOT 1, SPLINE PLOT 2, and DATA POINT PLOT

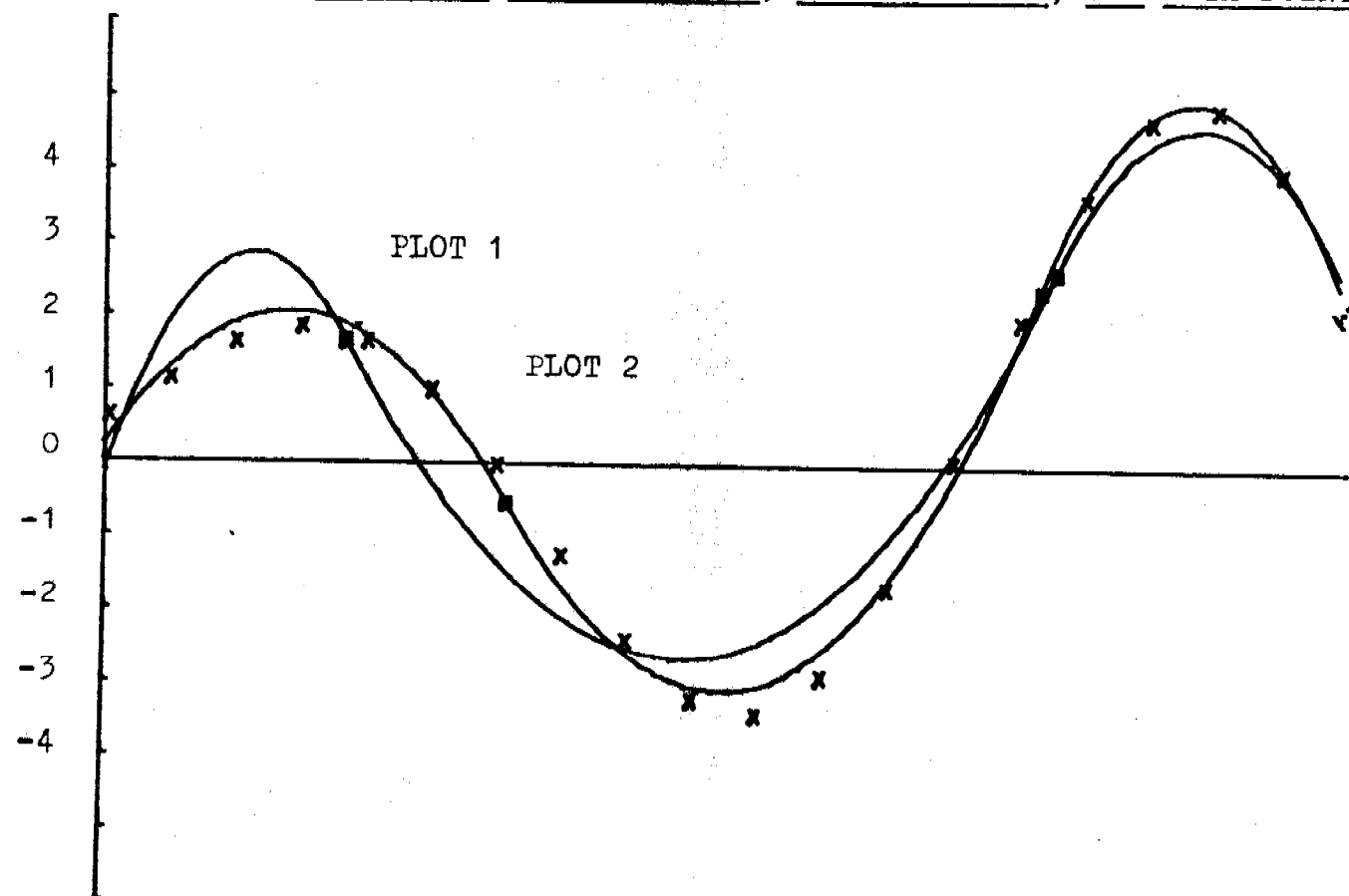


Figure 2b: ERROR PLOT FOR DATA POINT VS SPLINE 2 VALUE

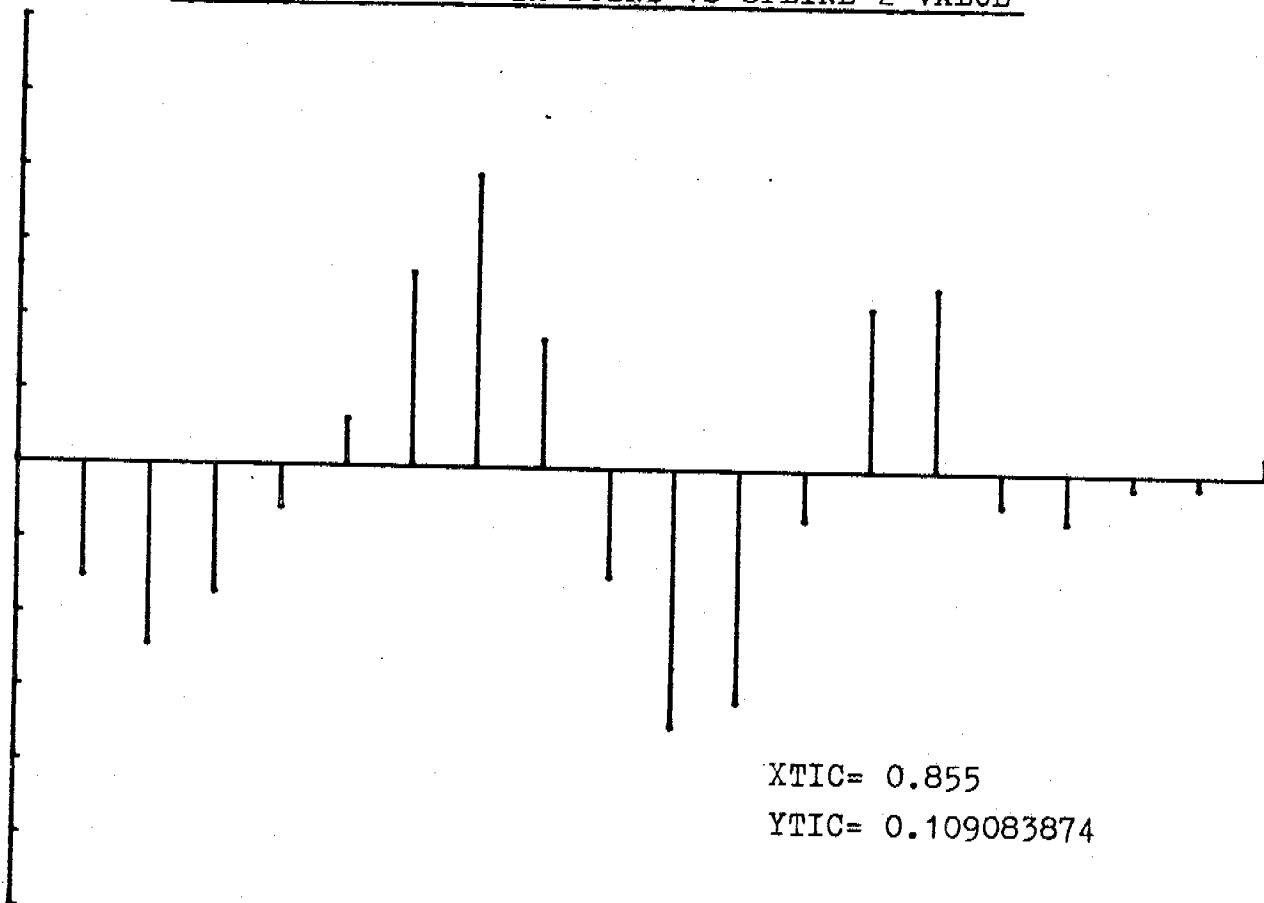


Figure 3: PRINTER OUTPUT FOR EXAMPLE 2

LINEAR SPLINE PROGRAM  
K EQUALS 2  
L9 EQUALS 20

N= 8

T(I) = 0	1
T(I) = 0	2
T(I) = 2.3	3
T(I) = 3.3	4
T(I) = 4.5	5
T(I) = 5.5	6
T(I) = 7.5	7
T(I) = 8.5	8
T(I) = 9.3	9
T(I) = 9.3	10

SUM OF SQUARES = 0.508496390

2.312506664	0.507836829
3.3	0.507836829
4.530897345	0.494072159
5.514309408	0.487454070
7.508981167	0.484908316
8.468580843	0.450663854

2.312506664	0.450663854
3.3	0.450663854
4.562157505	0.437636837
5.524269999	0.434330461
7.508981167	0.434330461
8.439467487	0.408240592

2.312506664	0.408240592
3.3	0.408240592
4.590545267	0.398339792
5.532607619	0.395976608
7.508981167	0.395976608
8.41376275	0.3778665422

Figure 4a: Example 2 LINEAR SPLINE PLOT AND DATA POINT PLOT

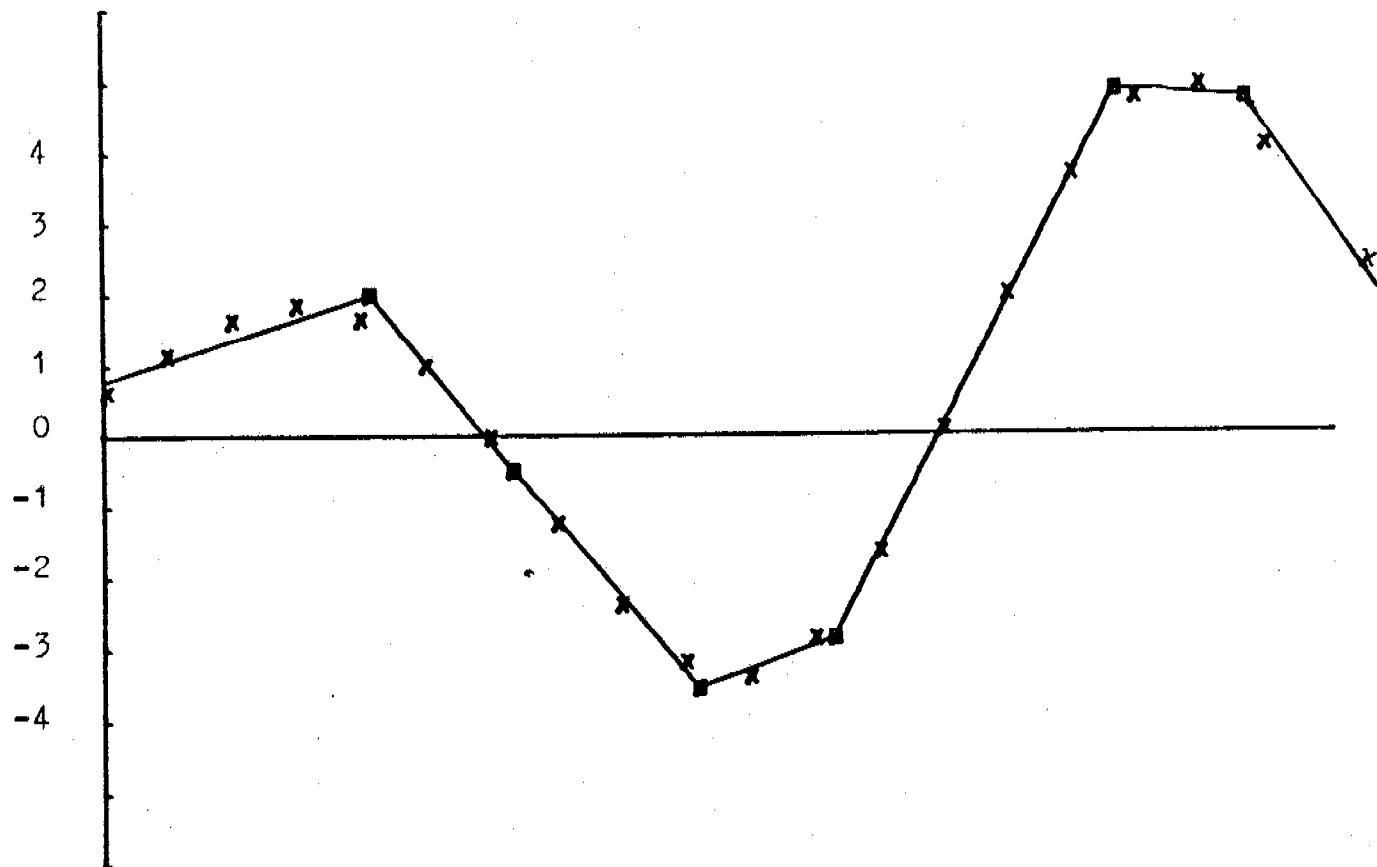


Figure 4b: ERROR PLOT FOR DATA POINT VS SPLINE VALUE

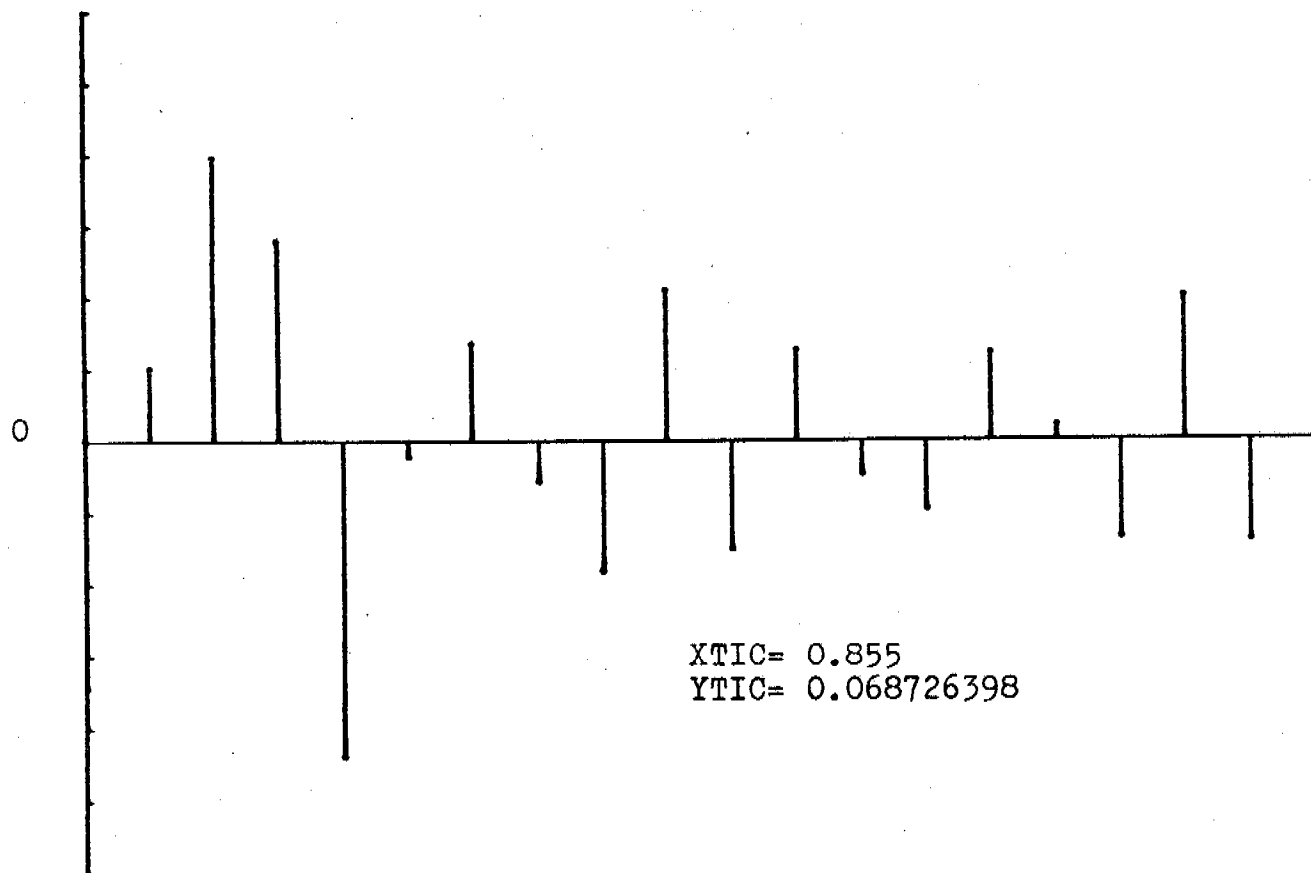


Figure 5: EXAMPLE OF OUTPUT FROM DERIVATIVE ROUTINE FOR  
20 EVENLY SPACED POINTS

<u>X-VALUE</u>	<u>SPLINE VALUE</u>	<u>FIRST DERIVATIVE</u>	<u>SECOND DERIVATIVE</u>
0	-1.276456667	3.851459424	-2.240811742
0.465	0.272212206	2.809481963	-2.240811742
0.93	1.336361559	1.767504503	-2.240811742
1.395	1.915991394	0.725527043	-2.240811742
1.86	2.011101709	-0.316450417	-2.240811742
2.325	1.621692505	-1.358427878	-2.240811742
2.79	0.747763783	-2.400405338	-2.240811742
3.255	-0.610305260	-3.383637786	2.309543558
3.72	-1.934006303	-2.309700031	2.309543558
4.185	-2.758326289	-1.235762277	2.309543558
4.65	-3.08326522	-0.161824522	2.309543558
5.115	-2.908823095	0.912113232	2.309543558
5.58	-2.234999914	1.986050987	2.309543558
6.045	-1.061795678	3.05998874	2.309543558
6.51	0.610789615	4.133926495	2.309543558
6.975	2.759180793	4.62786348	-4.825100395
7.44	4.389483645	2.384191797	-4.825100395
7.905	4.976479164	0.140520113	-4.825100395
8.37	4.520167350	-2.103151571	-4.825100395
8.835	3.020548203	-4.346823255	-4.825100395
9.3	0.477621723	-6.590494939	-4.825100395

```

10 REM **** LINEAR SPLINE PROGRAM ****
20 REM **** SUBROUTINE JUMP ADDRESSES ****
30 REM      (1) SPLINE GRAPH ----- 7000
40 REM      (2) ERROR GRAPH ----- 7380
50 REM      (3) DATA POINT PLOT ----- 7290
60 REM      (4) RETURN TO KNOT MOVING ROUTINE --- 7640
70 REM      (5) DERIVATIVE ROUTINE ----- 9000
80 DIM B[30],E[30],X[50],G[50],H$[1]
90 DIM N[6,6],T[30],P[30],M[30],D[6,6],A[30],C[30],S[6]
100 K=2
110 REM **** H8 USED IN KNOT MOVING ROUTINE ***
120 H8=128
130 PRINT "LINEAR S LINE PROGRAM"
140 PRINT "K EQUALS"K
150 DISP "NUMBER OF DATA PTS";
160 INPUT L9
170 PRINT "L9 EQUALS"L9
180 DISP "WANT TO LOAD DATA FROM TAPE";
190 INPUT H$
200 IF H$="Y" THEN 300
210 DISP "2 FILE #S WHERE DATA GOES";
220 INPUT A8,A9
230 FOR I=1 TO L9
  40 DISP "INPUT X,Y" I;
250 INPUT X[I],G[I]
260 NEXT I
270 STORE DATA A8,X
280 STORE DATA A9,G
290 GOTO 370
300 DISP "2 FILE #S WHERE DATA IS";
310 INPUT A8,A9
320 LOAD DATA A8,X
330 LOAD DATA A9,G
340 DISP "PRINT DATA";
350 INPUT H$
360 IF H$#"Y" THEN 400
370 FOR I=1 TO L9
380 PRINT I,X[I],G[I]
390 NEXT I
400 DISP "NUMBER OF EQUATIONS=?";
410 INPUT N
420 N1=N-1
430 PRINT "N="N
440 FOR I=1 TO N+K
450 DISP "KNOT IN T(I)=?" I;
460 INPUT T[I]
470 PRINT "T(I)="T[I],I
480 NEXT I
90 GOSUB 550
500 GOSUB 8070
510 PRINT "SUM OF SQUARES ="S1
520 DISP "PLOT?";
530 AIT 3000
540 GOTO 7640

```

```

550 REM *****SUBROUTINE EQUATE*****
570 FOR I1=1 TO N
580 B[I1]=C[I1]-P[I1]=0
610 NEXT I1
620 I=K
630 I1=0
640 FOR L=1 TO L9
650 IF I=N THEN 700
660 IF X[L]<T[I+1] THEN 700
670 I=I+1
680 I1=I-K
690 GOTO 650
700 T=X[L]
710 P1=T[I+1]-T
720 M1=T-T[I]
730 Z9=1/(P1+M1)
740 S1=P1*Z9
750 S2=M1*Z9
760 G1=G[L]
770 I2=I1+1
780 B[I2]=S1*G1+B[I2]
790 C[I2]=S1*S1+C[I2]
800 P[I2]=S1*S2+P[I2]
810 I2=I2+1
820 B[I2]=S2*G1+B[I2]
830 C[I2]=S2*S2+C[I2]
840 NEXT L
850 REM ***** BANDED MATRIX ROUTINE *****
860 REM SOLVES SYSTEM USING CHOLESKY'S DECOMPOSITION METHOD
870 FOR J=1 TO N1
880 J1=J+1
890 E[J]=D=SQR(C[J])
900 P[J]=D1=P[J]/D
910 C[J1]=C[J1]-D1*D1
920 NEXT J
930 E[N]=SQR(C[N])
940 C[1]=B[1]/E[1]
950 FOR J=2 TO N
960 J1=J-1
970 C[J]=(B[J]-P[J1]*C[J1])/E[J]
980 NEXT J
990 A[N]=C[N]/E[N]
1000 FOR J=N1 TO 1 STEP -1
1010 J1=J+1
1020 A[J]=(C[J]-P[J]*A[J1])/E[J]
1030 NEXT J
1040 RETURN

```



```

7000 REM*****SPLINE GRAPH*****
7010 DISP "4 SCALE VALUES";
7020 INPUT Q9,Q8,Q7,Q6
7030 SCALE Q9,Q8,Q7,Q6
7040 DISP "2 VALUES X-CROSS AND XTIC";
7050 INPUT Q7,Q6
7060 XAXIS Q7,Q6
7070 DISP "2 VALUES FOR Y-CROSS AND YTIC";
7080 INPUT Q7,Q6
7090 YAXIS Q7,Q6
7110 I=K
7120 FOR J=2 TO N+1
7130 T=T[J]
7140 GOSUB 8930
7150 GOSUB 7550
7160 PLOT T,C1
7170 NEXT J
7180 I=K
7190 FOR Q6=K+1 TO N
7200 PEN
7210 T=T[Q6]
7220 GOSUB 8930
7230 GOSUB 7550
7240 PLOT T,C1
7250 CPLOT -0.3,-0.3
7260 LABEL (*)"0"
7270 NEXT Q6
7280 STOP
7290 REM ***** DATA POINT PLOT *****
7300 FOR I1=1 TO L9
7310 PLOT X[I1],G[I1]
7320 CPLOT -0.3,-0.3
7330 LABEL (*)"X"
7350 PEN
7360 NEXT I1
7370 STOP
7380 REM ***** ERROR PLOT ROUTINE *****
7390 Y=3*SQR(S1/L9)
7400 SCALE X[1],X[L9],-Y,Y
7410 XAXIS 0,(X[L9]-X[1])/10
7420 YAXIS X[1],Y/6
7430 PRINT "XTIC ="(X[L9]-X[1])/10"YTIC="Y/6
7440 I=K
7450 FOR J=1 TO L9
7460 T=X[J]
7470 GOSUB 8930
7480 GOSUB 7550
7490 PLOT T,G[J]-C1
7495 PLOT T,0
7500 PEN
7510 NEXT J
7520 STOP

```

```

7530 REM ***** CALCULATE SPLINE VALUE FROM SPLINE COEFFICIENTS *****
7550 P1=T[I+1]-T
7570 M2=T-T[I]
7600 C1=(M2*A[I]+P1*A[I-1])/(M2+P1)
7630 RETURN
7640 REM*****KNOT MOVING SUBROUTINE*****
7650 FOR I5=3 TO N
7660 E1=(T[I5+1]-T[I5])/H8
7670 A5=S1
7680 X5=T[I5]
7690 T[I5]=X4=T[I5]+E1
7700 GOSUB 8070
7710 IF S1<A5 THEN 7770
7720 E1=(T[I5-1]-X5)/H8
7730 T[I5]=X4=X5+E1
7740 GOSUB 8070
7750 IF S1<A5 THEN 7770
7760 GOTO 7990
7770 A4=S1
7780 E1=E1+E1
7790 T[I5]=X3=X4+E1
7800 IF (X3<T[I5-1] OR X3>T[I5+1]) THEN 7990
7810 GOSUB 8070
7820 A3=S1
7830 IF (A3>A4) THEN 7890
7840 A5=A4
7850 X5=X4
7860 A4=A3
7870 X4=X3
7880 GOTO 7780
7890 X6=X3*X3
7900 X7=X4*X4
7910 X8=X5*X5
7920 R1=A5*(X6-X7)+A4*(X8-X6)+A3*(X7-X8)
7930 R2=(A5*(X3-X4)+A4*(X5-X3)+A3*(X4-X5))*2
7940 T[I5]=R1/R2
7950 GOSUB 8070
7960 IF (S1<A4) THEN 8000
7970 T[I5]=X4
7980 GOT 8000
7990 T[I5]=T[I5]-E1
8000 GOSUB 550
8010 GOSUB 8070
8020 PRINT T[I5],S1
8030 NEXT I5
8040 DISP "PLOT?";
8050 WAIT 3000
8060 GOTO 7650

```

```

8070 REM*****COMPUTES SUM OF SQUARES*****
8080 S1=0
8090 I=K
8100 FOR L=1 TO L9
8110 T=X[L]
8120 IF T[I+1] >= T THEN 8170
8130 I=I+1
8140 IF I <= N+1 THEN 8120
8150 PRINT "RANGE"
8160 STOP
8170 P1=T[I+1]-T
8190 M2=T-T[I]
8220 C1=(M2*A[I]+P1*A[I-1])/(M2+P1)
8250 S=C1-G[L]
8260 S1=S1+S*S
8270 NEXT L
8280 RETURN
8930 REM*** FIN S I SUCH THAT T(I)<=T<T(I+1) USING FORWARD SEARCH ONLY**
8940 IF T[I+1] >= T THEN 8990
8950 I=I+1
8960 IF I <= N+1 THEN 8940
8970 PRINT "RANGE";
8980 STOP
8990 RETURN

```

```

9000 REM ***** DERIVATIVES*****
9010 REM THIS ROUTINE CALCULATES THE ZEROth THROUGH D1 DERIVATIVES OF
9020 REM AND PLACES THEM IN P(1) THROUGH p(D1+1) RESP.          SPLIN
9030 REM THE DERIVATIVES ARE AT T
9040 D1=3
9050 H1=(T[N+1]-T[K])/60
9060 FOR T=T[K] TO T[N+1] STEP H1
9070 I=K
9080 GOSUB 8930
9090 REM***** CALCULATES N(I,K) AT T*****
9100 N[1,1]=1
9110 FOR S=1 TO K-1
9120 P[S]=T[I+S]-T
9130 M[S]=T-T[I+1-S]
9140 N[1,S+1]=0
9150 FOR R=1 TO S
9160 Z9=N[R,S]/(P[R]+M[S+1-R])
9170 N[R,S+1]=N[R,S+1]+P[R]*Z9
9180 N[R+1,S+1]=M[S+1-R]*Z9
9190 NEXT R
9200 NEXT S
9210 FOR S=1 TO K
9220 D[S,1]=A[I-S+1]
9230 NEXT S
9240 FOR S=2 TO K
9250 FOR R=S TO K
9260 D[R,S]=(D[R-1,S-1]-D[R,S-1])/(T[I+K-R+1]-T[I-R+S])
9270 NEXT R
9280 NEXT S
9290 M1=1
9300 FOR I=1 TO D1+1
9310 S1=0
9320 FOR J=1 TO K-I+1
9330 S1=S1+D[K+1-J,I]*N[J,K+1-I]
9340 NEXT J
9350 P[I]=S1*M1
9360 M1=M1*(K-I)
9370 NEXT I
9380 PRINT T,P[1],P[2],P[3]
9390 NEXT T
9400 STOP
9500 REM *** SPLINE FUNCTION FNS(T) *****
9510 DEF FNS(T)
9520 I=K
9530 GOSUB 8930
9540 GOSUB 7550
9550 RETURN C1
9560 END

```

```

10 REM***** QUADRATIC SPLINE PROGRAM ****
20 REM ***** SUBROUTINE JUMP ADDRESSES *****
30 REM      (1) SPLINE GRAPH ----- 7000
40 REM      (2) ERROR GRAPH ----- 7380
50 REM      (3) DATA POINT PLOT ----- 7290
60 R M      (4) RETURN TO KNOT MOVING ROUTINE --- 7640
70 REM      (5) DERIVATIVE ROUTINE ----- 9000
80 DIM B[30],E[30],X[50],G[50],H$[1]
90 DIM N[6,6],T[30],P[30],M[30],D[6,6],A[30],C[30],S[6]
100 K=3
110 REM*** H8 USED IN KNOT MOVING ROUTINE ***
120 H8=128
130 PRINT "QUADRATIC SPLINE PROGRAM"
140 PRINT "K EQUALS K"
150 DISP "NUMBER OF DATA PTS";
160 INPUT L9
17 PRINT "L9 EQUALS"L9
180 DISP "WANT TO LOAD DATA FROM TAPE";
190 INPUT H$
200 IF H$="Y" THEN 300
210 DISP "2 FILE #S WHERE DATA GOES";
220 INPUT A8,A9
230 FOR I=1 TO L9
40 DISP " INPUT X,Y" I;
250 INPUT X[I],G[I]
260 NEXT I
270 STORE DATA A8,X
280 STORE DATA A9,G
290 GOTO 370
300 DISP "2 FILE #S WHERE DATA IS";
310 INPUT A8,A9
30 LOAD DATA A8,X
330 LOAD DATA A9,G
340 DISP "PRINT DATA";
350 INPUT H$
360 IF H$#"Y" THEN 400
370 FOR I=1 TO L9
380 PRINT I,X[I],G[I]
390 NEXT I
400 DISP "NUMBER OF EQUATIONS=?";
410 INPUT N
420 N1=N-1
425 N2=N-2
430 PRINT "N="N
440 FOR I=1 TO N+K
450 DISP "KNOT IN T(I)=?" I;
460 INPUT T[I]
470 PRINT "T(I)=",T[I],I
480 NEXT I
490 GOSUB 550
500 GOSUB 8070
510 PRINT "SUM OF SQUARES ="S1
520 DISP "PLOT?";
530 WAIT 3000
540 GOTO 7640

```

```

550 REM *****SUBROUTINE EQUATE*****
560 FOR I1=1 TO N
570 B[I1]=C[I1]=P[I1]=M[I1]=0
610 NEXT I1
620 I=K
630 I1=0
640 FOR L=1 TO L9
650 IF I=N THEN 700
660 IF X[L]<T[I+1] THEN 700
670 I=I+1
680 I1=I-K
690 GOTO 650
700 T=X[L]
710 P1=T[I+1]-T
720 P2=T[I+2]-T
730 M1=T-T[I]
740 M2=T-T[I-1]
750 Z9=1/(P1+M1)
760 S1=P1*Z9
770 C2=M1*Z9
780 S2=C2
790 Z9=S1/(P1+M2)
800 S1=P1*Z9
810 C2=M2*Z9
820 Z9=S2/(P2+M1)
830 S2=C2+P2*Z9
840 S3=M1*Z9
850 G1=G[L]
860 I2=I1+1
870 B[I2]=S1*G1+B[I2]
880 C[I2]=S1*S1+C[I2]
890 P[I2]=S1*S2+P[I2]
900 M[I2]=S1*S3+M[I2]
910 I2=I2+1
920 B[I2]=S2*G1+B[I2]
930 C[I2]=S2*S2+C[I2]
940 P[I2]=S2*S3+P[I2]
950 I2=I2+1
960 B[I2]=S3*G1+B[I2]
970 C[I2]=S3*S3+C[I2]
980 NEXT L

```

```

990 REM ***** BANDED MATRIX ROUT NE *****
1000 REM SOLVES SYSTEM USING CHOLESKY'S DECOMPOSITION METHOD
1010 FOR J=1 TO N2
1020 J1=J+1
1030 J2=J+
1040 E[J]=D=SQR(C[J])
1050 P[J]=D1=P[J]/D
1060 M[J]=D2=M[J]/D
1070 C[J1]=C[J1]-D1*D1
1080 P[J1]=P[J1]-D1*D2
1090 C[J2]=C[J2]-D2*D2
1100 NEXT J
1110 E[N1]=D=SQR(C[N1])
1120 P[N1]=D1=P[N1]/D
1130 E[N]=SQR(C[N]-D1*D1)
1140 C[1]=B[1]/E[1]
1150 C[2]=(B[2]-C[1]*P[1])/E[2]
1160 FOR J=3 TO N
1170 J1=J-1
1180 J2=J-2
1190 C[J]=(B[J]-M[J2]*C[J2]-P[J1]*C[J1])/E[J]
1200 NEXT J
1210 A[N]=C[N]/E[N]
1220 A[N1]=(C[N1]-P[N1]*A[N])/E[N1]
1230 FOR J=N2 TO 1 STEP -1
1240 J1=J+1
1250 J2=J+2
1260 A[J]=(C[J]-P[J]*A[J1]-M[J]*A[J2])/E[J]
1270 NEXT J
1280 RETURN

```

```

7000 REM*****SPLINE GRAPH*****
7010 DISP "4 SCALE VALUES";
7020 INPUT Q9,Q8,Q7,Q6
7030 SCALE Q9,Q8,Q7,Q6
7040 DISP "2 VALUES X-CROSS AND XTIC";
7050 INPUT Q7,Q6
7060 XAXIS Q7,Q6
7070 DISP "2 VALUES FOR Y-CROSS AND YTIC";
7080 INPUT Q7,Q6
7090 YAXIS Q7,Q6
7100 DISP "STEP =";
7110 INPUT Q5
7120 I=K
7130 FOR T=Q9 TO Q8 STEP Q5
7140 GOSUB 8930
7150 GOSUB 7550
7160 PLOT T,C1
7170 NEXT T
7180 I=K
7190 FOR Q6=K+1 TO N
7200 PEN
7210 T=T[Q6]
7220 GOSUB 8930
7230 GOSUB 7550
7240 PLOT T,C1
7250 CPLOT -0.3,-0.3
7260 LABEL (*)"O"
7270 NEXT Q6
7280 STOP
7290 REM ***** DATA POINT PLOT *****
7300 FOR I1=1 TO L9
7310 PLOT X[I1],G[I1]
7320 CPLOT -0.3,-0.3
7330 LABEL (*)"X"
7350 PEN
7360 NEXT I1
7370 STOP
7380 REM ***** ERROR PLOT ROUTINE *****
7390 Y=3*SQR(S1/L9)
7400 SCALE X[1],X[L9],-Y,Y
7410 XAXIS 0,(X[L9]-X[1])/10
7420 YAXIS X[1],Y/6
7430 PRINT "XTIC="(X[L9]-X[1])/10"YTIC="Y/6
7440 I=K
7450 FOR J=1 TO L9
7460 T=X[J]
7470 GOSUB 8930
7480 GOSUB 7550
7490 PLOT T,G[J]-C1
7495 PLOT T,0
7500 PEN
7510 NEXT J
7520 STOP

```



```

7530 REM ***** CALCULATE SPLINE VALUE FROM SPLINE COEFFICIENTS *****
7550 P1=T[I+1]-T
7560 P2=T[I+2]-T
7570 M2=T-T[I-1]
7580 M3=T-T[I]
7590 C2=A[I-1]
7600 C1=(M2*C2+P1*A[I-2])/(M2+P1)
7610 C2=(M3*A[I]+P2*C2)/(M3+P2)
7620 C1=(M3*C2+P1*C1)/(M3+P1)
7630 RETURN
7640 REM*****KNOT MOVING SUBROUTINE*****
7650 FOR I5=4 TO N
7660 E1=(T[I5+1]-T[I5])/H8
7670 A5=S1
7680 X5=T[I5]
7690 T[I5]=X4=T[I5]+E1
7700 GOSUB 8070
7710 IF S1<A5 THEN 7770
7720 E1=(T[I5-1]-X5)/H8
7730 T[I5]=X4=X5+E1
7740 GOSUB 8070
7750 IF S1<A5 THEN 7770
7760 GOTO 7990
7770 A4=S1
7780 E1=E1+E1
7790 T[I5]=X3=X4+E1
7800 IF (X3<T[I5-1] OR X3>T[I5+1]) THEN 7990
7810 GOSUB 8070
7820 A3=S1
7830 IF (A3>A4) THEN 7890
7840 A5=A4
7850 X5=X4
7860 A4=A3
7870 X4=X3
7880 GOTO 7780
7890 X6=X3*X3
7900 X7=X4*X4
7910 X8=X5*X5
7920 R1= 5*(X6-X7)+A4*(X8-X6)+A3*(X7-X8)
7930 R2=(A5*(X3-X4)+A4*(X5-X3)+A3*(X4-X5))*2
7940 T[I5]=R1/R2
7950 GOSUB 8070
7960 IF (S1<A4) THEN 8000
7970 T[I5]=X4
7980 GOTO 8000
7990 T[I5]=T[I5]-E1
8000 GOSUB 550
8010 GOSUB 8070
8020 PRINT T[I5],S1
8030 NEXT I5
8040 DISP "PLOT?";
8050 WAIT 3000
8060 GOTO 7650

```

```

8070 REM*****COMPUTES SUM OF SQUARES*****
8080 S1=0
8090 I=K
8100 FOR L=1 TO L9
  110 T=X[L]
8120 IF T[I+1] >= T THEN 8170
8130 I=I+1
8140 IF I <= N+1 THEN 8120
8150 PRINT "OUT OF RANGE";
8160 STOP
8170 P1=T[I+1]-T
8180 P2=T[I+2]-T
8190 M2=T-T[I-1]
8200 M3=T-T[I]
8210 C2=A[I-1]
8220 C1=(M2*C2+P1*A[I-2])/(M2+P1)
8230 C2=(M3*A[I]+P2*C2)/(M3+P2)
8240 C1=(M3*C2+P1*C1)/(M3+P1)
8250 S=C1-G[L]
8260 S1=S1+S*S
8270 NEXT L
8280 RETURN
8930 REM** FINDS I SUCH THAT T(I)<=T<T(I+1) USING FORWARD SEARCH ONLY **
8940 IF T[I+1] >= T THEN 8990
8950 I=I+1
8960 IF I <= N+1 THEN 8940
8970 PRINT "RANGE";
8980 STOP
8990 RETURN

```

```

9000 REM ***** DERIVATIVES*****
9010 REM THIS ROUTINE CALCULATES THE ZERO TH THROUGH D1 DERIVATIVES OF SPLI
9020 REM AND PLACES THEM IN P(1) THROUGH p(D1+1) RESP.
9030 REM THE DERIVATIVES ARE AT T
9040 D1=3
9050 H1=(T[N+1]-T[K])/60
9060 FOR T=T[K] TO T[N+1] STEP H1
9070 I=K
9080 GOSUB 8930
9090 REM***** CALCULATES N(I,K) AT T*****
9100 N[1,1]=1
9110 FOR S=1 TO K-1
9120 P[S]=T[I+S]-T
9130 M[S]=T-T[I+1-S]
9140 N[1,S+1]=0
9150 FOR R=1 TO S
9160 Z9=N[R,S]/(P[R]+M[S+1-R])
9170 N[R,S+1]=N[R,S+1]+P[R]*Z9
9180 N[R+1,S+1]=M[S+1-R]*Z9
9190 NEXT R
9200 NEXT S
9210 FOR S=1 TO K
9220 D[S,1]=A[I-S+1]
9230 NEXT S
9240 FOR S=2 TO K
9250 FOR R=S TO K
9260 D[R,S]=(D[R-1,S-1]-D[R,S-1])/(T[I+K-R+1]-T[I-R+S])
9270 NEXT R
9280 NEXT S
9290 M1=1
9300 FOR I=1 TO D1+1
9310 S1=0
9320 FOR J=1 TO K-I+1
9330 S1=S1+D[K+1-J,I]*N[J,K+1-I]
9340 NEXT J
9350 P[I]=S1*M1
9360 M1=M1*(K-I)
9370 NEXT I
9380 PRINT T,P[1],P[2],P[3]
9390 NEXT T
9400 STOP
9500 REM *** SPLINE FUNCTION FNS(T) *****
9510 DEF FNS(T)
9520 I=K
9530 GOSUB 8930
9540 GOSUB 7550
9550 RETURN C1
9560 END

```

```

10 REM***** CUBIC SPLINE PROGRAM *****
20 REM **** SUBROUTINE JUMP ADDRESSES *****
30 REM      (1) SPLINE GRAPH ----- 7000
40 REM      (2) ERROR GRAPH ----- 7380
50 REM      (3) DATA POINT PLOT ----- 7290
60 REM      (4) RETURN TO KNOT MOVING ROUTINE --- 7640
70 REM      (5) DERIVATIVE ROUTINE ----- 9000
80 DIM B[30],E[30],X[50],G[50],H$[1],Q[30]
90 DIM N[6,6],T[30],P[30],M[30],D[6,6],A[30],C[30],S[6]
100 K=4
110 REM*** H8 USED IN KNOT MOVING ROUTINE ***
120 H8=128
130 PRINT "CUBIC SPLINE PROGRAM"
140 PRINT "K EQUALS"K
150 DISP "NUMBER OF DATA PTS";
160 INPUT L9
170 PRINT "L9 EQUALS"L9
180 DISP "WANT TO LOAD DATA FROM TAPE";
190 INPUT H$
200 IF H$="Y" THEN 300
210 DISP "2 FILE #S WHERE DATA GOES";
220 INPUT A8,A9
230 FOR I=1 TO L9
240 DISP "INPUT X,Y" I;
250 INPUT X[I],G[I]
260 NEXT I
270 STORE DATA A8,X
280 STORE DATA A9,G
290 GOTO 370
300 DISP "2 FILE #S WHERE DATA IS";
310 INPUT A8,A9
320 LOAD DATA A8,X
330 LOAD DATA A9,G
340 DISP "PRINT DATA";
350 INPUT H$
360 IF H$#"Y" THEN 400
370 FOR I=1 TO L9
380 PRINT I,X[I],G[I]
390 NEXT I
400 DISP "NUMBER OF EQUATIONS=?";
410 INPUT N
420 N1=N-1
425 N2=N-2
430 PRINT "N="N
440 FOR I=1 TO N+K
450 DISP "KNOT IN T(I)="? I;
460 INPUT T[I]
470 PRINT "T(I)="T[I],I
480 NEXT I
490 GOSUB 550
500 GOSUB 8100
510 PRINT "SUM OF SQUARES ="S1
520 DISP "PLOT?";
530 WAIT 3000
540 GOTO 7640

```

```

550 REM *****SUBROUTINE EQUATE*****
560 FOR I1=1 TO N
570 B[I1]=C[I1]=P[I1]=M[I1]=Q[I1]=0
620 NEXT I1
630 I=K
640 I1=0
650 FOR L=1 TO L9
660 IF I=N THEN 710
670 IF X[L]<T[I+1] THEN 710
680 I=I+1
690 I1=I-K
700 GOTO 660
710 T=X[L]
720 P1=T[I+1]-T
730 P2=T[I+2]-T
740 P3=T[I+3]-T
750 M1=T-T[I]
760 M2=T-T[I-1]
770 M3=T-T[I-2]
780 Z9=1/(P1+M1)
790 S1=P1*Z9
800 S2=M1*Z9
810 Z9=S1/(P1+M2)
820 S1=P1*Z9
830 C2=M2*Z9
840 Z9=S2/(P2+M1)
850 S2=C2+P2*Z9
860 S3=M1*Z9
870 Z9=S1/(P1+M3)
880 S1=P1*Z9
890 C2=M3*Z9
900 Z9=S2/(P2+M2)
910 S2=C2+P2*Z9
920 C3=M2*Z9
930 Z9=S3/(P3+M1)
940 S3=C3+P3*Z9
950 S4=M1*Z9
960 G1=G[L]
970 I2=I1+1
980 B[I2]=S1*G1+B[I2]
990 C[I2]=S1*S1+C[I2]
1000 P[I2]=S1*S2+P[I2]
1010 M[I2]=S1*S3+M[I2]
1020 Q[I2]=S1*S4+Q[I2]
1030 I2=I2+1
1040 B[I2]=S2*G1+B[I2]
1050 C[I2]=S2*S2+C[I2]
1060 P[I2]=S2*S3+P[I2]
1070 M[I2]=S2*S4+M[I2]
1080 I2=I2+1
1090 B[I2]=S3*G1+B[I2]
1100 C[I2]=S3*S3+C[I2]
1110 P[I2]=S3*S4+P[I2]
1120 I2=I2+1
1130 B[I2]=S4*G1+B[I2]
1140 C[I2]=S4*S4+C[I2]
1150 NEXT L

```

```

1160 REM ***** BANDED MATRIX ROUTINE *****
1170 REM SOLVES SYSTEM USING CHOLESKY'S DECOMPOSITION METHOD
1180 FOR J=1 TO N-3
1190 J1=J+1
1200 J2=J+2
1210 J3=J+3
1220 E[J]=D=SQR(C[J])
1230 P[J]=D1=P[J]/D
1240 M[J]=D2=M[J]/D
1250 Q[J]=D3=Q[J]/D
1260 C[J1]=C[J1]-D1*D1
1270 P[J1]=P[J1]-D1*D2
1280 M[J1]=M[J1]-D1*D3
1290 C[J2]=C[J2]-D2*D2
1300 P[J2]=P[J2]-D2*D3
1310 C[J3]=C[J3]-D3*D3
1320 NEXT J
1340 D=E[N2]=SQR(C[N2])
1350 P[N2]=D1=P[N2]/D
1360 M[N2]=D2=M[N2]/D
1370 C[N1]=C[N1]-D1*D1
1380 P[N1]=P[N1]-D1*D2
1390 C[N]=C[N]-D2*D2
1400 E[N1]=D=SQR(C[N1])
1410 P[N1]=D1=P[N1]/D
1420 E[N]=SQR(C[N]-D1*D1)
1430 C[1]=B[1]/E[1]
1440 C[2]=(B[2]-C[1]*P[1])/E[2]
1450 C[3]=(B[3]-M[1]*C[1]-P[2]*C[2])/E[3]
1460 FOR J=4 TO N
1470 J1=J-1
1480 J2=J-2
1490 J3=J-3
1500 C[J]=(B[J]-M[J2]*C[J2]-P[J1]*C[J1]-Q[J3]*C[J3])/E[J]
1510 NEXT J
1520 A[N]=C[N]/E[N]
1530 A[N1]=(C[N1]-P[N1]*A[N])/E[N1]
1540 A[N2]=(C[N2]-P[N2]*A[N1]-M[N2]*A[N])/E[N2]
1550 FOR J=N-3 TO 1 STEP -1
1560 J1=J+1
1570 J2=J+2
1580 J3=J+3
1590 A[J]=(C[J]-P[J]*A[J1]-M[J]*A[J2]-Q[J]*A[J3])/E[J]
1600 NEXT J
1610 RETURN

```

```

7000 REM*****SPLINE GRAPH*****
7010 DISP "4 SCALE VALUES";
7020 INPUT Q9,Q8,Q7,Q6
7030 SCALE Q9,Q8,Q7,Q6
7040 DISP "2 VALUES X-CROSS AND XTIC";
7050 INPUT Q7,Q6
7060 XAXIS Q7,Q6
7070 DISP "2 VALUES FOR Y-CROSS AND YTIC";
7080 INPUT Q7,Q6
7090 YAXIS Q7,Q6
7100 DISP "STEP =";
7110 INPUT Q5
7120 I=K
7130 FOR T=Q9 TO Q8 STEP Q5
7140 GOSUB 8930
7150 GOSUB 7540
7160 PLOT T,C1
7170 NEXT T
7180 I=K
7190 FOR Q6=K+1 TO N
7200 PEN
7210 T=T[Q6]
7220 GOSUB 8930
7230 GOSUB 7540
7240 PLOT T,C1
7250 CPLOT -0.3,-0.3
7260 LABEL (*)"0"
7270 NEXT Q6
7280 STOP
7290 REM ***** DATA POINT PLOT *****
7300 FOR I1=1 TO L9
7310 PLOT X[I1],G[I1]
7320 CPLOT -0.3,-0.3
7330 LABEL (*)"X"
7350 PEN
7360 NEXT I1
7370 STOP
7380 REM ***** ERROR PLOT ROUTINE *****
7390 Y=3*SQR(S1/L9)
400 SCALE X[1],X[L9],-Y,Y
7410 XAXIS 0,(X[L9]-X[1])/10
7420 YAXIS X[1],Y/6
7430 PRINT "XTIC ="(X[L9]-X[1])/10"YTIC="Y/6
7440 I=K
7450 FOR J=1 TO L9
7460 T=X[J]
7470 GOSUB 8930
7480 GOSUB 7540
7490 PLOT T,G[J]-C1
7495 PLOT T,0
7500 PEN
7510 NEXT J
7520 STOP

```

```

7530 REM ***** CALCULATE SPLINE VALUE FROM SPLINE COEFFICIENTS *****
7540 P1=T[I+1]-T
7550 P2=T[I+2]-T
7560 P3=T[I+3]-T
7570 M2=T-T[I-2]
7580 M3=T-T[I-1]
7590 M4=T-T[I]
7600 C1=(M2*A[I-2]+P1*A[I-3])/(M2+P1)
7610 C2=(M3*A[I-1]+P2*A[I-2])/(M3+P2)
615 C3=(M4*A[I]+P3*A[I-1])/(M4+P3)
7620 C1=(M3*C2+P1*C1)/(M3+P1)
7625 C2=(M4*C3+P2*C2)/(M4+P2)
7630 C1=(M4*C2+P1*C1)/(M4+P1)
7635 RETURN
7640 REM*****KNOT MOVING SUBROUTINE*****
7680 FOR I5=5 TO N
7690 E1=(T[I5+1]-T[I5])/H8
7700 A5=S1
7710 X5=T[I5]
7720 T[I5]=X4=T[I5]+E1
7730 GOSUB 8100
7740 IF S1<A5 THEN 7800
7750 E1=(T[I5-1]-X5)/H8
7760 T[I5]=X4=X5+E1
7770 GOSUB 8100
7780 IF S1<A5 THEN 7800
7790 GOTO 8020
7800 A4=S1
7810 E1=E1+E1
7820 T[I5]=X3=X4+E1
7830 IF (X3<T[I5-1] OR X3>T[I5+1]) THEN 8020
7840 GOSUB 8100
7850 A3=S1
7860 IF (A3>A4) THEN 7920
7870 A5=A4
7880 X5=X4
7890 A4=A3
7900 X4=X3
7910 GOTO 7810
7920 X6=X3*X3
7930 X7=X4*X4
7940 X8=X5*X5
7950 R1=A5*(X6-X7)+A4*(X8-X6)+A3*(X7-X8)
7960 R2=(A5*(X3-X4)+A4*(X5-X3)+A3*(X4-X5))*2
7970 T[I5]=R1/R2
7980 GOSUB 8100
7990 IF (S1<A4) THEN 8030
8000 T[I5]=X4
8010 GOTO 8030
8020 T[I5]=T[I5]-E1
8030 GOSUB 550
8040 GOSUB 8100
8050 PRINT T[I5],S1
8060 NEXT I5
8070 DISP "PLOT?";
8080 WAIT 3000
8090 GOTO 7680

```



```

8100 REM*****COMPUTES SUM OF SQUARES*****
8110 S1=0
8120 I=K
8130 FOR L=1 TO L9
8140 T=X[L]
8150 IF T[I+1] >= T THEN 8200
8160 I=I+1
8170 IF I <= N+1 THEN 8150
8180 PRINT "RANGE"
8190 STOP
8200 GOSUB 7530
8210 S=C1-G[L]
8220 S1=S1+S*S
8230 NEXT L
8240 RETURN
8930 REM** FINDS I SUCH THAT T(I)<=T<T(I+1) USING FORWARD SEARCH ONLY**
8940 IF T[I+1] >= T THEN 8990
8950 I=I+1
8960 IF I <= N+1 THEN 8940
8970 PRINT "RANGE";
8980 STOP
8990 RETURN

```

```

9000 REM ***** DERIVATIVES*****
9010 REM THIS ROUTINE CALCULATES THE ZERO TH THROUGH D1 DERIVATIVES OF
9020 REM AND PLACES THEM IN P(1) THROUGH p(D1+1) RESP.          SPLIN
9030 REM THE DERIVATIVES ARE AT T
9040 D1=3
9050 H1=(T[N+1]-T[K])/60
9060 FOR T=T[K] TO T[N+1] STEP H1
9070 I=K
9080 GOSUB 8930
9090 REM***** CALCULATES N(I,K) AT T*****
9100 N[1,1]=1
9110 FOR S=1 TO K-1
9120 P[S]=T[I+S]-T
9130 M[S]=T-T[I+1-S]
9140 N[1,S+1]=0
9150 FOR R=1 TO S
9160 Z9=N[R,S]/(P[R]+M[S+1-R])
9170 N[R,S+1]=N[R,S+1]+P[R]*Z9
9180 N[R+1,S+1]=M[S+1-R]*Z9
9190 NEXT R
9200 NEXT S
9210 FOR S=1 TO K
9220 D[S,1]=A[I-S+1]
9230 NEXT S
9240 FOR S=2 TO K
9250 FOR R=S TO K
9260 D[R,S]=(D[R-1,S-1]-D[R,S-1])/(T[I+K-R+1]-T[I-R+S])
9270 NEXT R
9280 NEXT S
9290 M1=1
9300 FOR I=1 TO D1+1
9310 S1=0
9320 FOR J=1 TO K-I+1
9330 S1=S1+D[K+1-J,I]*N[J,K+1-I]
9340 NEXT J
9350 P[I]=S1*M1
9360 M1=M1*(K-I)
9370 NEXT I
9380 PRINT T,P[1],P[2],P[3]
9390 NEXT T
9500 REM *** SPLINE FUNCTION FNS(T) *****
9510 DEF FNS(T)
9520 I=K
9530 GOSUB 8930
9540 GOSUB 7540
9550 RETURN C1
9560 END

```

## SOME NOVEL ROOTFINDING METHODS

Charles E. Gray

Aeronutronic Ford Corporation  
Palo Alto, California

### ABSTRACT

This paper develops some methods for computing the real roots of a real valued function in a finite interval. This classical problem is treated by methods that are novel in the following respects: the convergence does not depend on the customary "initial guess;" one can conclude that the function has no roots, if that be the case; if that is not the case, one can compute all real roots of the function in the interval to within a preassigned accuracy. It is assumed that the function is Lipschitz and that a value for the Lipschitz constant is known. These assumptions are mild enough so that methods are applicable to a variety of problems, notably in optimal control. From the basic method two other algorithms are derived: one computes the maximum of a function on an interval; the other computes the roots of a function, but with improved speed of convergence similar to that of Newton's method. Numerical results are presented which illustrate the properties discussed above.

## Introduction

A classical problem in numerical analysis is that of finding in a given interval (which we assume to be closed and finite) a real root of a continuous, real-valued function. Many algorithms have been proposed to solve this problem  $\left([1], [2], [3], [4], [5], [6]\right)$ .

In one way or another, they all require an initial guess at the value of the root; it is hoped that this guess is improved by an iterative application of the algorithm. This hope may be disappointed for a number of reasons. It may happen that an initial guess appropriate to the algorithm does not exist, even though a root exists. Or, if an initial guess and a root both exist, the algorithm may nevertheless fail to converge, because the initial guess is not close enough to the root. Of course, if there are no roots, no convergence is possible. Unfortunately, it is not possible to determine whether lack of convergence is due to the first or the second of these alternatives. The only recourse then is: if at first you don't converge, pick a new initial guess and try, try again.

By contrast, the method which we present below (first proposed in [7]) does not require an initial guess at the value of the root. If there is no root of the function in the given interval, the algorithm reveals this fact. Otherwise, the algorithm computes all roots of the function in the interval.

To illustrate the problems that can beset classical methods, we consider two of them: the bisection method and Newton's method. We denote the function whose roots are sought by  $f(x)$  and the interval of interest by  $[a, b]$ . The problem then is to find an  $x \in [a, b]$  such that

$$f(x) = 0$$

### The Bisection Method

In the bisection method it is assumed that we know two numbers  $u_0 \in [a, b]$  and  $v_0 \in [a, b]$ , such that  $f(u_0)$  and  $f(v_0)$  are of opposite sign. Since  $f(x)$  is continuous, there is at least one  $x \in [u_0, v_0]$  such that  $f(x) = 0$ . (See Figure 1.) The bisection method will compute one root, as follows:

Step 0     Guess  $u_0, v_0$  such that  $f(u_0) \cdot f(v_0) < 0$ . Set  $i = 0$

Step 1     Set  $x_i = 1/2 (u_i + v_i)$

        - If  $f(x_i) \cdot f(u_i) < 0$  set  $v_{i+1} \leftarrow x_i$

            Go to Step 2

        - If  $f(x_i) \cdot f(u_i) = 0$  Stop:  $x_i$  is the solution.

        - If  $f(x_i) \cdot f(u_i) > 0$  Set  $u_{i+1} \leftarrow x_i$

            Go to Step 2

Step 2     Set  $i \leftarrow i+1$  and go to Step 1

However, it may happen that there are no  $u_0$  and  $v_0$ ; in the example of Figure 2,  $f(x) \geq 0 \forall x \in [a, b]$  hence  $f(u_0) \cdot f(v_0) \geq 0$   
 $\forall u_0 \in [a, b], v_0 \in [a, b]$

### Newton's Method

In Newton's method, it is assumed that an initial guess  $x_0$  is given. This guess is then updated according to the formula

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Like the little girl, who had a little curl, right in the middle of her forehead, when Newton's method is good, it is very, very good, but when it is bad it is horrid (See Figures 4, 5, 6).

### A New Root Finding Method.

#### Concept

Suppose that  $f(x)$  is Lipschitz on  $[a,b]$ . Thus for some  $M'$ :

$$|f(x_1) - f(x_2)| < M' |x_1 - x_2| \quad \forall x_1, x_2 \in [a,b]$$

Assume that  $M'$  is known and consider the following algorithm:

#### Algorithm

Step 0 Set  $i \leftarrow 0$ ,  $x_i \leftarrow a$ .

Step 1 Compute  $f(x_i)$ .

Go to Step 2.

Step 2 Set  $x_i + \frac{|f(x_i)|}{M'} \rightarrow x_{i+1}$

Go to Step 3

Step 3 If  $x_i > b$ , stop.

Otherwise set  $i \leftarrow i+1$  and go to Step 1.

The geometric interpretation of the algorithm is the following (cf. Figure 7):

To obtain  $x_{i+1}$ , draw from the point with coordinates  $(x_i, f(x_i))$  the line  $L$  whose slope is  $M'$  in magnitude and which intersects the  $x$ -axis to the right of  $x_i$ ; the point of intersection is  $x_{i+1}$ .

#### Convergence Properties

The algorithm generates a monotonically increasing sequence  $\{x_i\}$ . If this sequence is finite, then the algorithm stops in Step 3 with  $x_i > b$ .

If the sequence is infinite  $\{x_i\}_{i=0}^{\infty}$  then it has a limit in  $[a,b]$ , say  $x^*$  (because  $\{x_i\}_{i=0}^{\infty}$  is a sequence bounded from above by  $b$ ).

Then:

$$\lim_{i \rightarrow \infty} x_{i+1} = \lim_{i \rightarrow \infty} x_i = x^*$$

Because  $f(x)$  is Lipschitz, it is continuous so that

$$\lim_{i \rightarrow \infty} |f(x_i)| = |f(x^*)|$$

From Step 2 of the algorithm we conclude

$$x^* = x^* + \frac{1}{M'} |f(x^*)|$$

Hence:

$$f(x^*) = 0$$

Thus the algorithm either converges to a root in an infinite number of steps or it reaches the end of the interval ( $x_i > b$ ) in a finite number of steps without computing a root. Note that this conclusion holds for any positive value of  $M'$ .

But  $M'$  is a Lipschitz constant. This key fact allows us to strengthen our conclusions, which we now state in the form of a theorem.

#### Theorem

Let  $f(x)$  be a real valued function on  $[a,b]$  such that

$$|f(x_1) - f(x_2)| < M' |x_1 - x_2| \quad \forall x_1, x_2 \in [a,b]$$

Consider a sequence  $\{x_i\}_{i=0}^{\infty}$  generated by applying Algorithm 1 to  $f(x)$ .

Then either

- (1) The sequence is finite, in which case  $f(x)$  has no roots in  $[a,b]$

or

- (2) The sequence is infinite, in which case it converges to the smallest  $x \in [a,b]$  such that  $f(x) = 0$ .

#### Geometrical Sketch of Proof

Refer to Figure 7. Since  $M'$  is a Lipschitz constant, it is greater than the magnitude of the slope of  $f(x)$  anywhere on  $[a,b]$ . Hence the

line  $L$ , so to speak, goes to zero faster than the function. Consequently if  $f(x_i) > 0$ , then  $f(x) > 0 \forall x \in [x_i, x_{i+1}]$ . Therefore if  $f(a) > 0$  then  $f(x) > 0 \forall x \in [a, x_i]$  for every  $i = 1, 2, \dots$ . If the sequence  $x_i$  "jams up" at  $x^* \in [a, b]$  then  $f(x^*) = 0$  while  $f(x) > 0 \forall x \in [a, x^*)$ . Otherwise,  $f(x) > 0 \forall x \in [a, b]$  as in Figure 8.

For a formal proof see the appendix.

## Realization

### Introduction

Algorithm 1 allows us to determine the first root of  $f(x)$  to the right of  $x = a$ , say  $x_1^*$ ; by applying Algorithm 1 again we can compute the first root to the right of  $x_1^*$ , say  $x_2^*$ , i.e., the second root to the right of  $x = a$  and so on until all the roots of  $f(x)$  in  $[a, b]$  have been computed.

Note that, as stated, Algorithm 1 requires in general an infinite number of steps to compute  $x_1^*$ , and therefore will not be able to compute  $x_2^*$ , in finite time. In order to complete the computation in finite time, we must modify Algorithm 1; the result is Algorithm 2, which we call ROOT-FINDER.

Algorithm 2 is obtained by relaxing the requirement that we compute the set  $S_0 = \{x \in [a, b] \mid f(x) = 0\}$  to the requirement that we compute some set  $S_\epsilon$  of the form  $S_\epsilon = \{x \in [a, b] \mid |f(x)| \leq \epsilon\}$  where  $\epsilon$  is a preassigned number. Clearly  $S_\epsilon$  contains  $S_0$ . Also, as  $\epsilon \rightarrow 0$ ,  $S_\epsilon \rightarrow S_0$  so that  $S_\epsilon$  is an approximation to the set of roots of  $f(x)$ ,  $S_0$ ; the closeness of the approximation  $\epsilon$ , can be preassigned, (See Figure 11).



Algorithm 2: ROOTFINDER

- Step 0 Set  $N_\epsilon \leftarrow 1$ ,  $i \leftarrow 0$ ,  $N_R \leftarrow 0$ ,  $x_i \leftarrow a$ ,  $j \leftarrow 1$ . Let  $\epsilon > 0$  be given
- Step 1 Compute  $f(x_i)$
- Step 2 If  $|f(x_i)| < \epsilon$  go to Step 3;  
Otherwise go to Step 6.
- Step 3 If  $N_\epsilon = 1$  go to Step 4;  
Otherwise go to Step 5.
- Step 4 Set  $\xi_j \leftarrow x_i$ , go to Step 5.
- Step 5 Set  $N_\epsilon \leftarrow 0$ ,  $h \leftarrow \frac{\epsilon}{M'}$ ,  $N_R \leftarrow 1$   
Go to Step 9.
- Step 6 If  $N_\epsilon = 1$ , go to Step 3;  
Otherwise go to Step 7.
- Step 7 Set  $\eta_j \leftarrow x_{i-1}$ ,  $j \leftarrow j+1$   
Go to Step 8.
- Step 8 Set  $N_\epsilon \leftarrow 1$ ,  $h \leftarrow \frac{|f(x_i)|}{M'}$ .  
Go to Step 9.
- Step 9 Set  $x_{i+1} \leftarrow x_i + h$   
 $i \leftarrow i + 1$
- Step 10 If  $x_i < b$  go to Step 1;  
Otherwise go to Step 11.
- Step 11 If  $N_\epsilon = 1$  go to Step 13;  
Otherwise go to Step 12.
- Step 12 Set  $b \leftarrow \eta_j$ ,  $j+1 \rightarrow j$ , go to Step 13.
- Step 13 If  $N_R = 0$ , report "There are no roots" and stop. Otherwise,  
set  $j \leftarrow j+1$ , report: "Roots lie in the intervals  $[\xi_k, \eta_k]_{k=1}^J$ "  
and stop.

### Discussion

Referring to the flow chart of Figure 9 we see that  $x_i$  is updated in one of two ways:

$$1) \text{ If } |f(x_i)| > \epsilon \quad \text{then} \\ x_{i+1} = x_i + \frac{|f(x_i)|}{M'}$$

as in Algorithm 1. The step size  $x_{i+1} - x_i$  is variable, being proportional to  $|f(x_i)|$ .

$$2) \text{ If } |f(x_i)| \leq \epsilon \quad \text{then} \\ x_{i+1} = x_i + \frac{\epsilon}{M'}$$

In this case the step size  $x_{i+1} - x_i$  is constant.

The flag  $N_\epsilon$  takes on the value 0 if at the previous  $x_i$ ,  $|f(x_i)| < \epsilon$  and takes on the value 1 otherwise.

To understand algorithm 2 refer to Figure 10 which depicts the function  $f(x)$  in the vicinity of a zero crossing. The path followed through the algorithm is analyzed in Table 1.

The key points are the following:

- 1) If  $N_\epsilon = 1$  then, because of the property of the variable step size of Algorithm 1 the function does not vanish in  $[x_{i-1}, x_i]$ .
- 2) If  $N_\epsilon = 0$ ,  $x_i$  is incremented by the constant step size which is chosen small enough so that in one step  $|f(x)|$  cannot exceed  $\epsilon$  (due to the bound on the slope of  $|f(x)|$ ).
- 3) If  $N_\epsilon = 1$  and  $|f(x)| < \epsilon$  the algorithm enters an interval  $[\xi_j, \eta_j]$  where by 2) above  $|f(x)|$  does not exceed  $\epsilon$ .
- 4) If  $N_\epsilon = 0$  and  $|f(x)| \geq \epsilon$  the algorithm leaves an interval  $[\xi_j, \eta_j]$  where  $|f(x)|$  does not exceed  $\epsilon$ .

In summary, Algorithm 2 computes a set of  $\epsilon$ -intervals  $\left\{ \left[ \xi_j, \eta_j \right] \right\}$  in which  $\left| f(x) \right|$  does not exceed  $\epsilon$  and outside of which (as shown in the discussion of Algorithm 1)  $\left| f(x) \right|$  does not vanish (See Figure 11). The flag  $N_R$  is initialized at 0; it is set to 1 when the first  $\epsilon$ -interval is entered (i.e., when  $x = \xi_1$ ). If when the algorithm stops  $N_R$  is 0, then at no point in  $[a, b]$  is  $\left| f(x) \right| < \epsilon$ , so that  $f(x)$  has no roots in  $[a, b]$ .

In terms of the discussion in the introduction we have

$$S_\epsilon = \bigcup [\xi_j, \eta_j]$$

Remark:

Considering the simplicity of the concept underlying ROOTFINDER and GNEWTON, it seems remarkable that these algorithms have remained undiscovered until now, yet such seems to be the case. This is not to say there haven't been some near misses.

Thus, for example, Milne [2] discusses algorithms of the form

$$x_{i+1} = x_i - \frac{1}{m} f(x_i)$$

Various choices for  $m$  are considered but the choice of a Lipschitz constant is not among them. Also, the use of the absolute value of the second term on the right hand side is not considered; this would ensure that the search always proceeds to the right, as in ROOTFINDER.

### Application

### Introduction

Algorithm 2 can be modified to find approximately the maximum of the function  $f(x)$  on  $[a,b]$ . The idea is to find the zeros of  $f(x)-c$ , where  $c$  is a constant; then, by some search procedure, increase  $c$  until the algorithm indicates that  $f(x)-c$  is non-positive but not strictly negative on  $[a,b]$ .

Algorithm 3 which we call MAXFINDER described below, computes a value  $c + \epsilon$  which exceeds the maximum of  $f(x)$  on  $[a,b]$  by no more than  $\epsilon$  (where  $\epsilon$  is a preassigned number) together with intervals  $[\xi_j, \eta_j]$  where  $f(x)$  lies in the band  $c-\epsilon, c+\epsilon$  (See Figure 12).

Algorithm 3:    MAXFINDER

- Step 0     Set  $0 \rightarrow N_1$      $1 \rightarrow N_\epsilon$  .  $0 \rightarrow i$      $1 \rightarrow j$      $a \rightarrow x_i \rightarrow \xi_j$   
go to Step 1.
- Step 1     Compute  $f(x_i)$ . Go to Step 2.
- Step 2     If  $N_1=0$  go to Step 3.  
Otherwise go to Step 4.
- Step 3     Set  $f(x_i) \rightarrow c$      $1 \rightarrow N_1$  . Go to Step 4.
- Step 4     If  $f(x_i) < c + \epsilon$  Go to Step 8.  
Otherwise, go to Step 5.
- Step 5     Set  $f(x_i) \rightarrow c$      $1 \rightarrow j$ . Go to Step 6.
- Step 6     Set  $x_i \rightarrow \xi_j$ . Go to Step 7.
- Step 7     Set  $\frac{\epsilon}{M'} \rightarrow h$ ,     $0 \rightarrow N_\epsilon$ . Go to Step 13.
- Step 8     If  $f(x_i) > c - \epsilon$ , go to Step 9.  
Otherwise, go to Step 10.
- Step 9     If  $N_\epsilon = 0$ , go to Step 7.  
Otherwise go to Step 6.
- Step 10    If  $N_\epsilon = 0$ , go to Step 11.  
Otherwise go to Step 12.
- Step 11    Set  $1 \rightarrow N_\epsilon$  ,     $x_{i-1} \rightarrow \eta_j$ ,     $j+1 \rightarrow j$   
Go to Step 12.
- Step 12    Set  $\frac{c-f}{M'} \rightarrow h$ . Go to Step 13.
- Step 13    Set  $x_i + h \rightarrow x_{i+1}$  ,     $i+1 \rightarrow i$  Go to Step 14.
- Step 14    If  $x_i > b$  , go to Step 15. Otherwise go to Step 1.
- Step 15    If  $N_\epsilon = 0$  , go to Step 16. Otherwise go to Step 17.

Step 16 Set  $b \rightarrow \eta_j$ ,  $j+1 \rightarrow j$ . Go to Step 17.

Step 17 Set  $j-1 \rightarrow J$ .

Report:  $c - \epsilon < \text{Max } f(x) < c + \epsilon$

$c - \epsilon < f(x) < c + \epsilon$  in intervals  $\left\{ \left[ \xi_k, \eta_k \right] \right\}_{k=1}^J$

Stop.

### Discussion

Algorithm 3 can be understood by referring to the flow chart of Figure 14 and to Figure 15, which represents a sample  $f(x)$  to be maximized.

As long as  $f(x)$  increases the algorithm steps along in steps of  $\frac{\epsilon}{M'}$ , updating  $c$  to  $f(x_i)$  whenever  $f(x_i)$  exceeds the previous  $c$  by more than  $\epsilon$ ; at the same time  $\xi_j$  is updated to  $x_i$ . When  $f(x_i)$  starts to decrease, the value of  $x_i$  at which the decrease is noted, i.e., the  $x_i$  previous to the one such that  $f(x_i) - c < -\epsilon$ , is retained as  $\eta_j$ . The current approximation to the maximum is then the current  $c$  while the location of the maximum is approximated by  $\left[ \xi_j, \eta_j \right]$ .

When  $f(x)$  decreases, the algorithm steps along in variable steps proportional to  $c - f(x)$  in the manner of Algorithm 1.

In Figure 15, successive approximations to the maximum are  $c^1, c^2, \dots, c^6$ . After  $c^6$  the function starts to decrease so  $c^6$  is taken to be  $c$ , the approximation to the maximum. Likewise,  $\xi_1^6$  is taken to be the approximation to the left hand end of the  $\epsilon$ -interval around the maximum. When  $f(x_i) < c - \epsilon$ ,  $x_{i-1}$  is taken to be  $\eta_1$ , and the algorithm starts stepping in variable steps proportional to  $c - f(x)$  in the manner of Algorithm 1. The procedure may be likened to plotting  $f(x)$  on the side of an aquarium, then

filling the aquarium with water until the maximum of the curve  $f(x)$  just touches the surface of the water. Notice that this is a direct search, which yields the global maximum. This is in contrast to the method of searching for the set of roots of the derivative of  $f(x)$ , which may or may not contain the global maximum (See Figure 13).

### Acceleration of Convergence

#### Introduction

One of the virtues of Newton's method is that it exhibits very rapid convergence when "close enough" to a root of  $f(x)$ . It is possible to combine Algorithm 2 and Algorithm 3 to give a new algorithm, Algorithm 4, which we call GNEWTON, which has the convergence properties of Newton's method in the vicinity of a root but, in the manner of Algorithm 2, computes all roots of  $f(x)$  in the interval  $[a,b]$  (or indicates that there are none) without requiring an initial guess. This, Newton's method cannot do.

Algorithm 4 is based on two ideas: the first is to introduce an iteration on  $\epsilon$ : the interval  $[a,b]$  is scanned by ROOTFINDER with a coarse value of  $\epsilon$ , ROOTFINDER computes  $\epsilon$ -intervals  $[x_1, y_1], \dots, [x_k, y_k]$ . Then  $\epsilon$  is refined so that, say, new  $\epsilon = \epsilon' < \text{old } \epsilon$ . As we saw above  $f(x) \neq 0$  outside of  $[x_1, y_1] \dots [x_k, y_k]$ , so on the next pass ROOTFINDER need only scan  $[x_1, y_1], \dots, [x_k, y_k]$ . The algorithm then computes  $\epsilon'$ -intervals  $[x'_1, y'_1], \dots, [x'_k, y'_k]$ . Again  $\epsilon$  is refined, the  $\epsilon'$  intervals are scanned and so on until  $\epsilon$  is less than some preassigned  $\epsilon^*$ . Significant improvements in computational efficiency result when this idea is combined with the following one: prior to scanning an  $\epsilon$ -interval  $[x_k, y_k]$ , with ROOTFINDER, MAXFINDER is used to estimate the maximum value of the slope of  $f(x)$  on  $[x_k, y_k]$ . The result is then used as  $M'$  in ROOTFINDER.



Algorithm 4: GNEWTON

Step 0 Set  $a \rightarrow x$ ,  $b \rightarrow y$ , Choose  $\epsilon^*$  and  $\epsilon > \epsilon^*$ . Set  $1 \rightarrow K \rightarrow k$ .

Step 1 Apply MAXFINDER to the function  $f'(x)$  on the interval  $[x, y]$  to compute  $M'$ .

Step 2 Using  $M'$  computed in Step 1, apply ROOTFINDER to  $f(x)$  on the interval  $[x, y]$

Store the resulting  $\epsilon$ -intervals  $\left\{ \left[ \xi_j, \eta_j \right] \right\}$  in Stack B (Number these sequentially as they are computed).

Step 3 If  $k < K$  go to Step 6. Otherwise go to Step 4.

Step 4 If  $\epsilon \leq \epsilon^*$  Report results.

Otherwise go to Step 5.

Step 5 Refine  $\epsilon$ .

Note  $K$ , the total number of  $\epsilon$ -intervals just computed

Transfer  $\left\{ \left[ \xi_k, \eta_k \right] \right\}$  the  $\epsilon$ -intervals just computed, from Stack B to Stack A, which contains the intervals to be scanned on the next pass; label them  $\left[ x_1, y_1 \right], \dots, \left[ x_K, y_K \right]$ .

Clear Stack B.

Set  $1 \rightarrow k$

Go to Step 7.

Step 6 Set  $k + 1 \rightarrow k$ . Go to Step 7.

Step 7 Set  $x_k \rightarrow x$   $y_k \rightarrow y$ .

Go to Step 1.

Discussion

Algorithm 4 can be understood by referring to the flowchart of Figure 16 and the discussion in the introduction. Some further discussion of the entries in Stack B is appropriate: the intervals  $\left[ \xi_j, \eta_j \right]$  are numbered sequentially in the order in which they are computed. For example, suppose that in scanning

$[x_1, y_1]$  three intervals are computed; these are numbered 1, 2, 3 from left to right. Then in scanning  $[x_2, y_2]$ , two more intervals are computed; these would then be numbered 4 and 5; and so on. An example is given in Figure 17.

### Convergence Properties

As mentioned before, when close enough to a root, Newton's method exhibits very rapid convergence. To see why this is so, let us consider Newton's iteration formula in the vicinity of a simple root  $x^*$ .

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Using Taylor's expansion around  $x^*$  for  $f(x_i)$  and  $f'(x_i)$  there follows:

$$x_{i+1} - x^* = x_i - x^* - \frac{f(x^*) + f'(x^*)(x_i - x^*) + o((x_i - x^*)^2)}{f'(x^*) + o(x_i - x^*)}$$

where  $o(\cdot)$  denotes any function such that:

$$\lim_{\alpha \rightarrow 0} \frac{o(\alpha)}{\alpha} < \infty$$

Noting that  $f(x^*) = 0$  and  $f'(x^*) \neq 0$  (because  $x^*$  is a simple root) there follows:

$$x_{i+1} - x^* = o((x_i - x^*)^2)$$

Hence the speed of convergence [ 7 ] of the method is at least quadratic.

Notice that the same result would be obtained with any iteration scheme of the form:

$$x_{i+1} = x_i - \frac{f(x_i)}{\phi(x_i)}$$

provided only that:

$$\phi(x_i) = f'(x^*) + o(x_i - x^*)$$

In fact, GNEWTON appears to be just such an iteration scheme. This is suggested by the following argument. Consider (for simplicity) the case of a single, simple root  $x^*$  of  $f(x)$ . Denote by  $[\xi^i, \eta^i]$  the successive  $\epsilon$ -intervals around it computed by GNEWTON and by  $M'(\xi^i, \eta^i)$  the corresponding value of  $M'$  computed by MAXFINDER. Then:

$$\text{Max } f'(x) - \epsilon < M'(\xi^i, \eta^i) < \text{Max } f'(x) + \epsilon$$

If the sequence  $\epsilon$  converges to 0, we obtain

$$M'(\xi^i, \eta^i) = |f'(x^*)| + o(\xi^i - x^*)$$

It then seems plausible to assert that, for small  $|\xi^i - x^*|$  the iteration scheme is of the form:

$$\xi^{i+1} = \xi^i + \frac{|f(\xi^i)|}{M'(\xi^i, \eta^i)} = \xi^i - \frac{f(\xi^i)}{f'(x^*) + o(\xi^i - x^*)}$$

which, by what was said above, assures Newton convergence. This conjecture is supported by the numerical behavior of GNEWTON, as will be discussed below.

## Discussion

### Comparison of assumptions

Let us compare first the assumptions made in using bisection and in using ROOTFINDER. In both cases  $f(x)$  is assumed to be continuous. In addition, ROOTFINDER assumes that the slope of  $f(x)$  is bounded and that a bound  $M'$  is known, while the bisection method assumes that  $f(x)$  changes sign on  $[a, b]$ .

Among the questions that we might ask at this point are the following. Are these assumptions unduly restrictive? Are they easy to verify? These are rather vague questions, so the answers will likewise be somewhat vague.

In many applications the functions of interest are continuous. The class of band limited functions, for example, are not only continuous but have bounded slope and the maximum value of the slope can be predicted on the basis of the bandwidth. In other cases, the assumptions can be verified by inspection of the analytical expression for  $f(x)$  together with a few rough numerical calculations.

It is worth noting that the assumptions underlying ROOTFINDER buy somewhat more than those underlying bisection, since ROOTFINDER provides information about all the roots of  $f(x)$ , including none if there are none.

If we now compare GNEWTON and NEWTON, we find that both assume that  $f(x)$  is continuously differentiable. In addition GNEWTON assumes that the slope of  $|f'(x)|$  is bounded and that a bound  $M''$  is known. This additional assumption buys, as above, information about all the roots of  $f(x)$ .

Since these assumptions are stronger than those made for ROOTFINDER or bisection, the class of functions to which GNEWTON or NEWTON can be applied is more restricted. This class is by no means empty, as the following non-trivial example will show.

In solving a certain linear optimal control problem ([7], [9]) it is necessary to find the roots in  $[a, b]$  of a function  $f(x)$  of the form

$$f(x) = \phi(x) - \beta$$

where  $\beta$  is a constant and  $\phi(x)$  is a given function of the form:

$$\phi(x) = \sum_{i=1}^n a_i e^{\lambda_i x} \quad (\lambda_i \text{ real, } \neq 0)$$

Thus, the derivative  $\phi'(x)$  is given by:

$$\phi'(x) = \sum_{i=1}^n a_i \lambda_i e^{\lambda_i x}$$

The upper bound  $M''$  on the slope of  $\phi'$  can be crudely estimated from the second derivative  $\phi''(x)$ :

$$\phi''(x) = \sum_{i=1}^n a_i \lambda_i^2 \frac{\lambda_i^x}{e}$$

Such an estimate is:

$$M'' = \sum_{i=1}^n \left| a_i \right| \lambda_i^2 \times \begin{cases} e^{\lambda_i^a} & \text{if } \lambda_i < 0 \\ \lambda_i^b & \text{if } \lambda_i > 0 \end{cases} \quad \begin{matrix} \text{(Note that} \\ a < b) \end{matrix}$$

This problem must be solved for a sequence of values of  $\beta$ , so that it is important to have the fastest convergence possible. It is also important to compute all the roots of  $f(x)$  in  $[a, b]$ . Thus, GNEWTON appears to be made to order for this application.

#### Comparison of Numerical Performance

Newton's method, GNEWTON and ROOTFINDER were applied to the problem of finding the roots of  $f(x)$  on  $[a, b]$  for the particular case where:

$$f(x) = k + \sin 2\pi x$$

$$[a, b] = [0, 1]$$

The constant  $k$  was chosen successively to be 0, 1 and 2, thus illustrating the case where  $f(x)$  has several simple roots, one double root and no roots at all (See Figure 18).

The termination criterion was

$$|f(x)| < \epsilon$$

Three values of  $\epsilon$  were used,  $10^{-2}$ ,  $10^{-4}$ ,  $10^{-6}$ .

Also, to illustrate the effect of overestimating  $M'$  (for ROOTFINDER) or  $M''$  (for GNEWTON) three values were considered:

$$M' = M'' = M_0 = 2 \pi$$

$$M' = M'' = 5 M_0$$

$$M' = M'' = 10 M_0$$

Here  $M_0$  is the optimum value of the Lipschitz constant  $M'$  or  $M''$ .

The measure of efficiency of a method was taken to be the total number of function evaluations of  $f(x)$  and  $f'(x)$  required to meet the termination criterion.

To compare two methods, the relative efficiency was computed by dividing the number of function evaluations for one method by the number of function evaluations for the other. If one method could not converge (as for example, when the bisection method was applied to a non-negative function) the number of steps was taken to be infinite.

The results are presented in Tables III, IV, V, VI, and VII. The data on which they are based is presented in Table II.

Table III compares the performance of GNEWTON and ROOTFINDER. Because of the overhead involved in evaluating the slope GNEWTON is less efficient than ROOTFINDER when  $M'$  is equal to the value of the slope of  $f(x)$  at the root ( $M' = 2 \pi, f(x) = \sin 2 \pi x$  i.e.  $k = 0$ ).

As  $M'$  increases ROOTFINDER takes smaller and smaller steps and its efficiency decreases. As  $M''$  increases the efficiency of the slope estimations in GNEWTON decreases because of the smaller steps; however, the improved slope estimates accelerate the zero searching phase of GNEWTON so that overall, its efficiency relative to ROOTFINDER increases.

Likewise, in the case ( $M' = 2 \pi, f(x) = 1 + \sin 2 \pi x$ , i. e.  $k = 1$ ) the value of  $M'$  is much greater than the value of the slope at the root (which

is 0). Thus ROOTFINDER must take small steps to avoid overstepping the root, while GNEWTON can adapt its step size. The result is that GNEWTON becomes significantly more efficient.

Tables IV and V show the degradation in efficiency of the algorithms as a function of  $M'$  or  $M''$  relative to their own performance when  $M'$  or  $M''$  are chosen optimally ( $M' = M'' = 2\pi$ ). The degradation of GNEWTON is slight, while ROOTFINDER degrades by an order of magnitude when  $M'$  is overestimated by an order of magnitude.

Table VI compares ROOTFINDER to NEWTON. When  $M'$  is well chosen ( $M' = 2\pi$ ) and the slope of the function near the root is close to  $M'$  ( $k = 0$ ) ROOTFINDER is only slightly less efficient than NEWTON. As these conditions are departed from, its efficiency decreases. (The table illustrates only the result of varying  $k$ ). However, for  $k > 1$  NEWTON does not converge while ROOTFINDER does, so its relative efficiency is infinite.

Table VI also compares ROOTFINDER to BISECTION. It shows that ROOTFINDER is uniformly more efficient (i.e. for every  $k$  and  $\epsilon$ , when  $M' = 2\pi$ ).

Table VII similarly compares GNEWTON to NEWTON and BISECTION. The method is somewhat less efficient than either of these methods except in the cases where they fail to converge ( $k = 1, 2$  for BISECTION,  $k = 2$  for NEWTON).

Finally Table VIII summarizes these results by tabulating the efficiencies of GNEWTON, NEWTON and BISECTION relative to that of ROOTFINDER for  $M' = 2\pi$  which was chosen because it converges in every case).

It is to be noted that these comparisons are overly favorable to NEWTON and BISECTION as they do not take into account the computational effort necessary to obtain an initial guess, which necessarily degrades the efficiency.

### Advantages and limitations of GNEWTON and ROOTFINDER

The distinguishing feature of GNEWTON and ROOTFINDER is that their convergence does not depend on an "initial guess". This is especially important if they are to be used as a subalgorithm within a larger program, where human supervision of a trial and error search is impractical.

The logic of ROOTFINDER is simple enough that it can be programmed on a commercially available 8 register, 49 program step pocket calculator with 19 steps and 4 registers left for programming the function whose roots are sought. This is sufficient to solve non-trivial problems. On the other hand it can, in some cases, be prohibitively inefficient. This shortcoming is overcome to a degree by GNEWTON at the cost of a somewhat more limited applicability and more algorithm complexity.

A second shortcoming of these methods is that it is difficult to extend them to several dimensions. Indeed, in order to preserve the feature that all roots are computed, the computational effort must increase at least exponentially with the dimension of the space. For the same reason it is difficult to extend MAXFINDER to many dimensions.

### Conclusions

We have developed some methods for computing the real roots of a real valued function in a finite interval. These methods are novel in that their convergence does not depend on the customary "initial guess"; one can conclude that the function has no roots, if that be the case; and if it is not, one can compute all the roots of the function in the interval. On the opposite side of the coin, these methods appear to be substantially less efficient than the notably efficient but erratic method of Newton. This penalty in efficiency may



be a reasonable price to pay for the definitive nature of the results obtained. This is particularly true if these algorithms are to be used as subalgorithms within a larger program, in which case human supervision of a trial and error search would not be practical.

#### References:

- [ 1 ] R. W. Hamming, "Numerical Methods for Scientists and Engineers" McGraw Hill 1962
- [ 2 ] W. E. Milne, "Numerical Calculus", Princeton University Press 1949
- [ 3 ] Wendell L. Grove, "Brief Numerical Methods" Prentice-Hall Inc.
- [ 4 ] A. S. Householder, "Principles of Numerical Analysis" McGraw-Hill, 1953
- [ 5 ] Forman S. Acton, "Numerical Methods that Work" Harper & Row 1970
- [ 6 ] Jon M. Smith, "Scientific Analysis on the Pocket Calculator" John Wiley 1975
- [ 7 ] C. E. Gray, "Optimal Control of Systems with combined pulse width and pulse frequency modulation". Doctoral Dissertation Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California, March 1971
- [ 8- ] J. F. Traub, "Iterative Methods for the Solution of Equations" Prentice-Hall Inc.
- [ 9 ] C. E. Gray, "Optimal Control of Systems Using N-State Controls" IEEE Transactions on Automatic Control Vol. AC-19, No. 4 August 1974

## APPENDIX

### Proof of Theorem

Suppose first that  $f(a) = 0$ . Then from Step 2 of Algorithm 1:

$$\{x_i\} = \{a, a, a, \dots\}$$

Suppose now that  $f(a) > 0$  (to fix ideas). For any  $x_i$ , if  $f(x_i) > 0$ , then  $f(x) > 0 \quad \forall x \in [x_i, x_{i+1}]$ .

Indeed from Step 2 of Algorithm 1:

$$M'(x_{i+1} - x_i) = f(x_i) > 0$$

From the Lipschitz condition

$$(1) \quad f(x) - f(x_i) > -M'(x - x_i) \geq -M(x_{i+1} - x_i) \quad \forall x \in [x_i, x_{i+1}]$$

Hence: 
$$f(x) > 0 \quad \forall x \in [x_i, x_{i+1}] \quad i = 0, 1, 2, \dots$$

Thus the sequence  $\{x_i\}$  is strictly monotonically increasing. If it is finite, it must be true that for some  $k$ ,  $x_k > b$  since only this condition leads to a stop command in the algorithm. In this case:

$$f(x) > 0 \quad \forall x \in [a, x_k]$$

Since  $[a, x_k] \supset [a, b]$ ,  $f(x)$  has no roots in  $[a, b]$ .

Conversely, if the sequence is infinite, then for every  $i$ :

$$x_i \leq b$$

Hence the sequence has a least upper bound  $x^* \leq b$ . Since the sequence is monotonic, it converges to  $x^*$ . Therefore, as was shown above,

$$f(x^*) = 0$$

On the other hand from (1) there follows:

$$f(x) > 0 \quad \forall x \in [a, x^*)$$

Hence  $x^*$  is the smallest  $x^*$  in  $[a, b]$  such that  $f(x^*) = 0$ .

TABLE I

x	CONDITIONS		PATH THROUGH ALGORITHM	REMARKS
	$N_\epsilon$	$ f(x) $		
$x_0$	1	$\geq \epsilon$	1-2-6-8-9-10-1	$N_\epsilon$ initialized to 1
$x_1$	1	$\geq \epsilon$	1-2-6-8-9-10-1	$N_\epsilon$ computed as 1
$x_2$	1	$< \epsilon$	1-2-3-4-5-9-10-1	Transition of $ f(x) $ from $\geq \epsilon$ to $< \epsilon$ $\xi_j$ is left hand end of interval where $ f(x)  < \epsilon$ . There is at least one such interval, hence $N_R$ is set to 0.
$x_3$	0	$< \epsilon$	1-2-3-5-9-10-1	$ f(x) $ remains $< \epsilon$
$x_4$	0	$\geq \epsilon$	1-2-6-7-8-9-10-1	Transition of $ f(x) $ from $< \epsilon$ to $\geq \epsilon$ $\eta_3$ is Right hand end of interval where $ f  < \epsilon$ . Set value of $j$ to $j + 1$ as index of next such interval.

TABLE II

Algorithm	k	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	Remarks
ROOTFINDER( $2\pi$ )	0	7.7	12.3	17	(*) (See below)
	1	56	565	5656	
	2	4	4	4	
ROOTFINDER( $10\pi$ )	0	42	72.7	103.3	(*) (See below)
	1	283	2825	28280	
	2	19	19	19	
ROOTFINDER( $20\pi$ )	0	84	145.3	206.7	(*) (See below)
	1	566	5650	56560	
	2	37	37	37	
GNEWTON( $2\pi$ )	0	18.7	25.7	33.3	(*) (See below)
	1	46	92	161	
	2	7	7	7	
GNEWTON( $10\pi$ )	0	22.3	30	37	(*) (See below)
	1	95	153	211	
	2	15	15	15	
GNEWTON ( $20\pi$ )	0	24.7	33	40	(*) 3 roots To get total function evaluations multiply entry by 3.
	1	115	218	325	
	2	18	18	18	
BISECTION	0	10	17	23	} BISECTION cannot converge
	1	$\infty$	$\infty$	$\infty$	
	2	$\infty$	$\infty$	$\infty$	
NEWTON	0	5	7	7	NEWTON cannot converge
	1	7	13	21	
	2	$\infty$	$\infty$	$\infty$	

Average Number of function evaluations per root for various algorithms.

TABLE III

Lipschitz Constant	k	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	REMARKS
2 $\pi$	0	.41	.48	.51	Optimum Lipschitz Constant
	1	1.22	6.14	35.13	
	2	.57	.57	.57	
10 $\pi$	0	1.88	2.42	2.79	5 x degraded Lipschitz Constant
	1	2.98	18.46	134.03	
	2	1.27	1.27	1.27	
20 $\pi$	0	3.4	4.4	5.17	10 x degraded Lipschitz constant
	1	4.92	25.92	174.03	
	2	2.06	2.06	2.06	

Efficiency of GNEWTON relative to ROOTFINDER as a function of Lipschitz constant

(M', M''), accuracy (  $\epsilon$  ) and function (  $f_k(x) = k + \sin 2\pi x$  ).

TABLE IV

Lipschitz Constant	k	<sup>-2</sup> $\epsilon = 10$	<sup>-4</sup> $\epsilon = 10$	<sup>-6</sup> $\epsilon = 10$	Remarks
$2\pi$	0	1	1	1	Optimum value of Lipschitz constant
	1	1	1	1	
	2	1	1	1	
$10\pi$	0	.84	.86	.90	
	1	.48	.60	.76	
	2	.47	.47	.47	
$20\pi$	0	.76	.78	.83	
	1	.40	.42	.50	
	2	.39	.39	.39	

Efficiency of ROOTFINDER (M') relative to ROOTFINDER ( $2\pi$ )

TABLE V

Lipschitz Constant	k	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	Remarks
$2 \pi$	0	1	1	1	Optimum value of Lipschitz constant
	1	1	1	1	
	2	1	1	1	
$10 \pi$	0	.18	.17	.16	
	1	.20	.20	.20	
	2	.21	.21	.21	
$20 \pi$	0	.09	.08	.08	
	1	.10	.10	.10	
	2	.11	.11	.11	

Efficiency of GNEWTON (M'') relative to GNEWTON ( $2 \pi$ )

TABLE VI

ROOTFINDER Compared to	k	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	Remarks
BISECTION	0	1.3	1.38	1.35	Optimum Lipschitz constant used for ROOTFINDER
	1	$\infty$	$\infty$	$\infty$	Bisection cannot converge
	2	$\infty$	$\infty$	$\infty$	
NEWTON	0	.65	.57	.41	Optimum Lipschitz constant used for ROOTFINDER
	1	.13	.02	.004	Newton cannot converge
	2	$\infty$	$\infty$	$\infty$	

Efficiency of ROOTFINDER relative to BISECTION and NEWTON.



TABLE VII

GNEWTON Compared to	k	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	Remarks
BISECTION	0	.54	.56	.69	Optimum Lipschitz constant used for GNEWTON
	1	$\infty$	$\infty$	$\infty$	BISECTION cannot converge
	2	$\infty$	$\infty$	$\infty$	
NEWTON	0	.26	.27	.21	Optimum Lipschitz Constant used for GNEWTON
	1	.15	.14	.13	NEWTON cannot converge
	2	$\infty$	$\infty$	$\infty$	

Efficiency of GNEWTON relative to BISECTION and NEWTON

TABLE VIII

Algorithm Compared to ROOTFINDER ( $2\pi$ )	k=0	k=1	k=2	Remarks
ROOTFINDER ( $2\pi$ )	1	1	1	ROOTFINDER's performance degrades by a factor of 10
ROOTFINDER ( $20\pi$ )	.09	.10	.11	
GNEWTON ( $2\pi$ )	.41	1.22	.57	GNEWTON's performance degrades by a factor less than 3
GNEWTON ( $20\pi$ )	.31	.49	.22	
BISECTION	.77	0	0	BISECTION cannot converge for $k \geq 1$
NEWTON	1.53	8	0	NEWTON cannot converge for $k = 2$

Efficiency of various algorithms relative to the optimum ROOTFINDER algorithm for  $\epsilon = 10^{-2}$

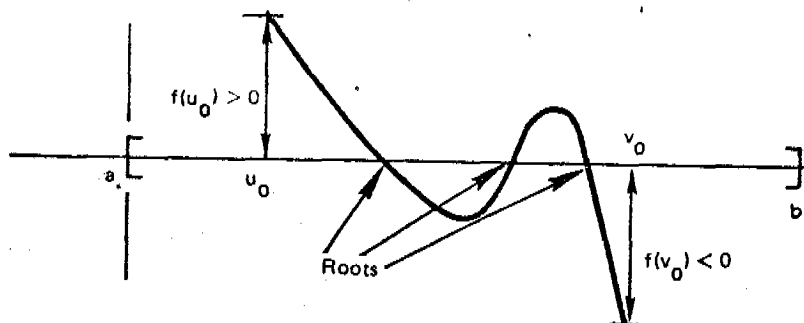


Figure 1. The bisection method works

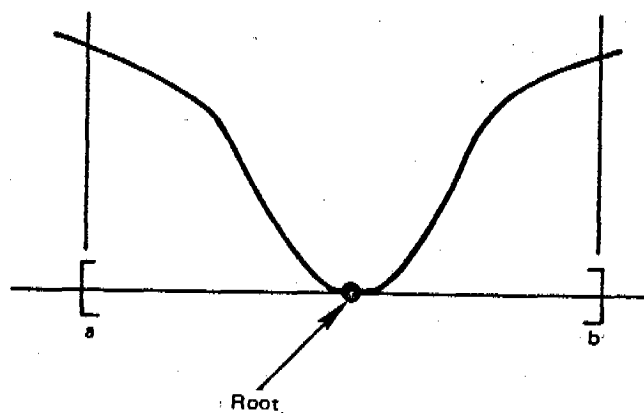


Figure 2. The bisection method fails  
(there do not exist  $u_0, v_0 \in [a, b]$  such that  $f(u_0)f(v_0) < 0$ )

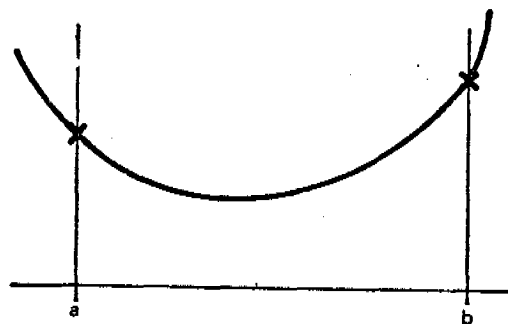


Figure 3. The bisection method fails  
(there is no root)

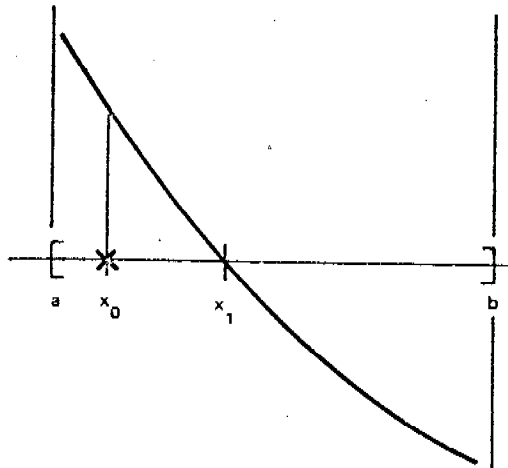


Figure 4. Newton's method is very, very good

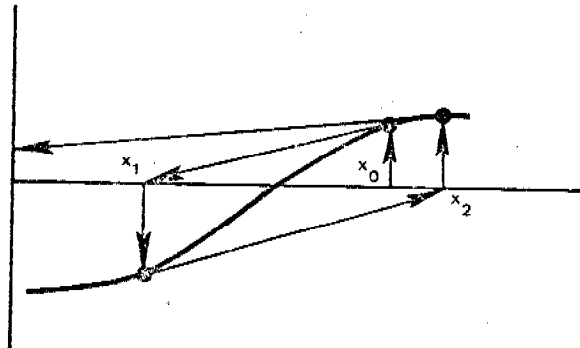


Figure 5. Newton's method is horrid  
(because of an inflection point near the root)

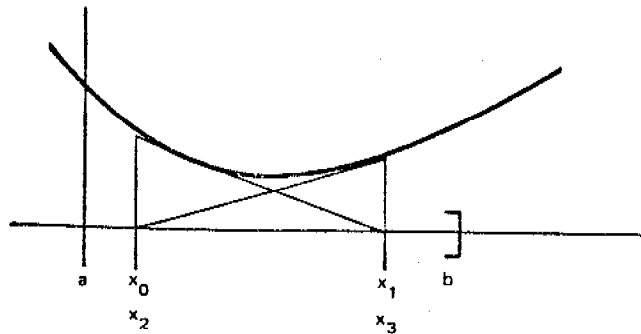


Figure 6. Newton's method is horrid  
(because there is no root)

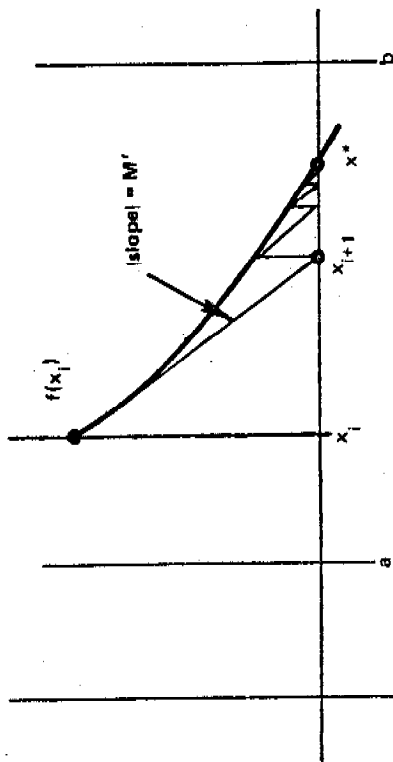


Figure 7.  $f(x)$  converges to  $x^*$

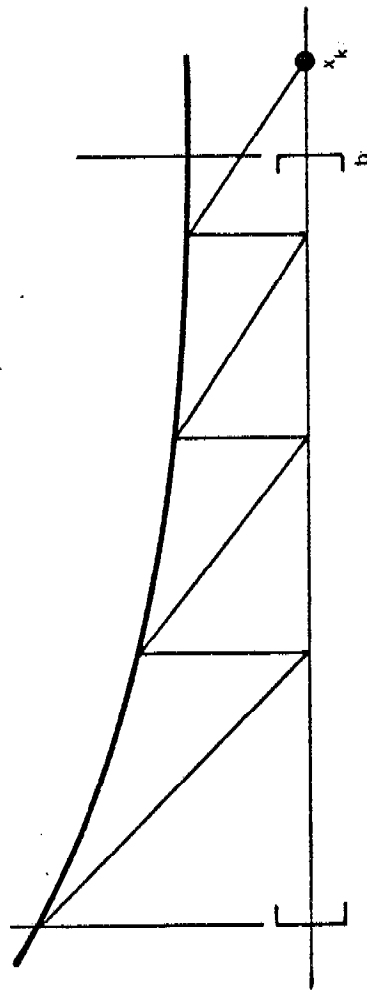


Figure 8.  $f(x)$  has no roots in  $[a, b]$

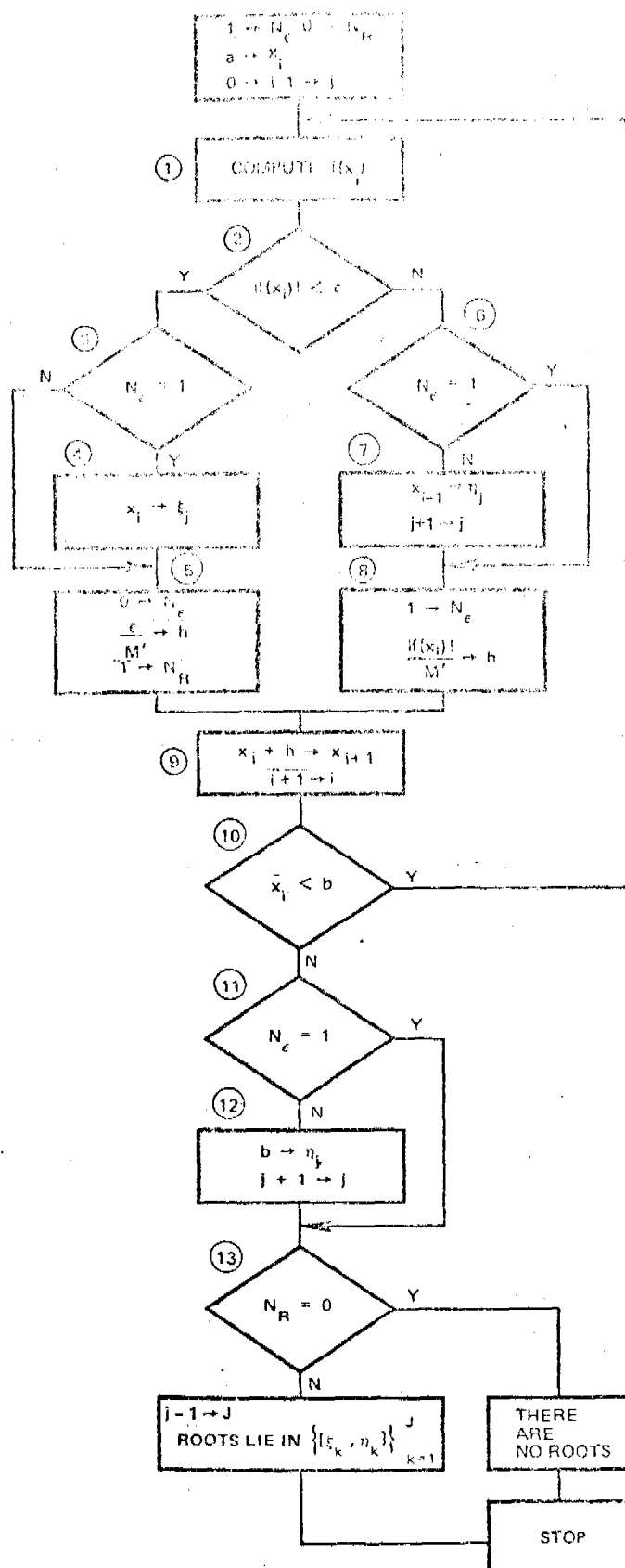


Figure 9. Flow chart of Algorithm 2

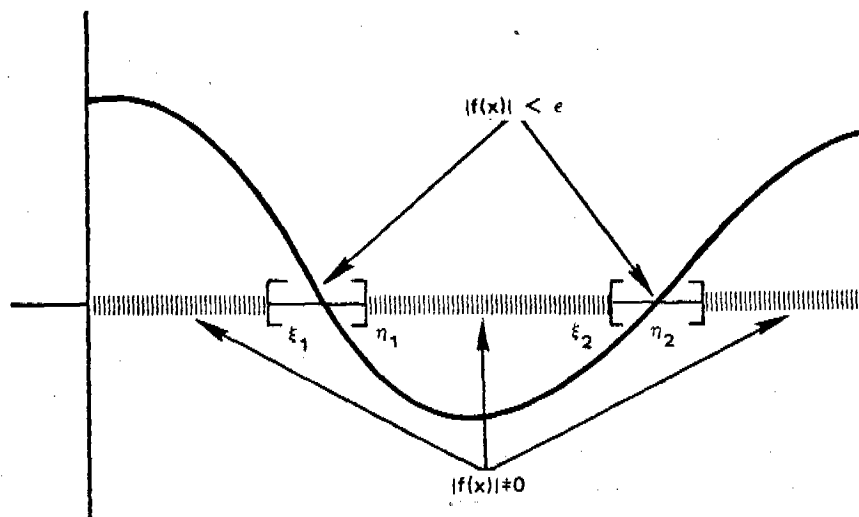


Figure 11. Example of a set  $S_\epsilon = \{(\xi_1, \eta_1), (\xi_2, \eta_2)\}$

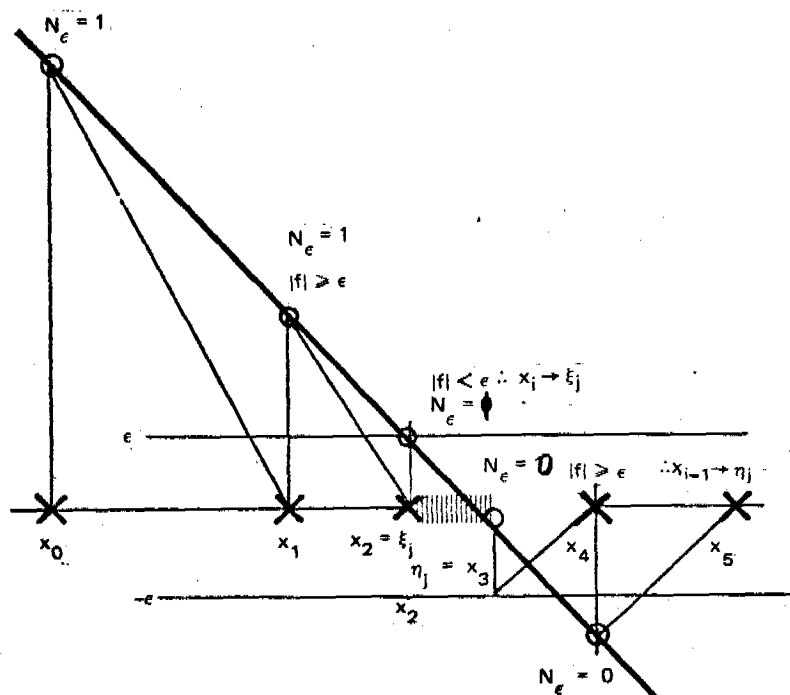


Figure 10. Analysis of Algorithm 2

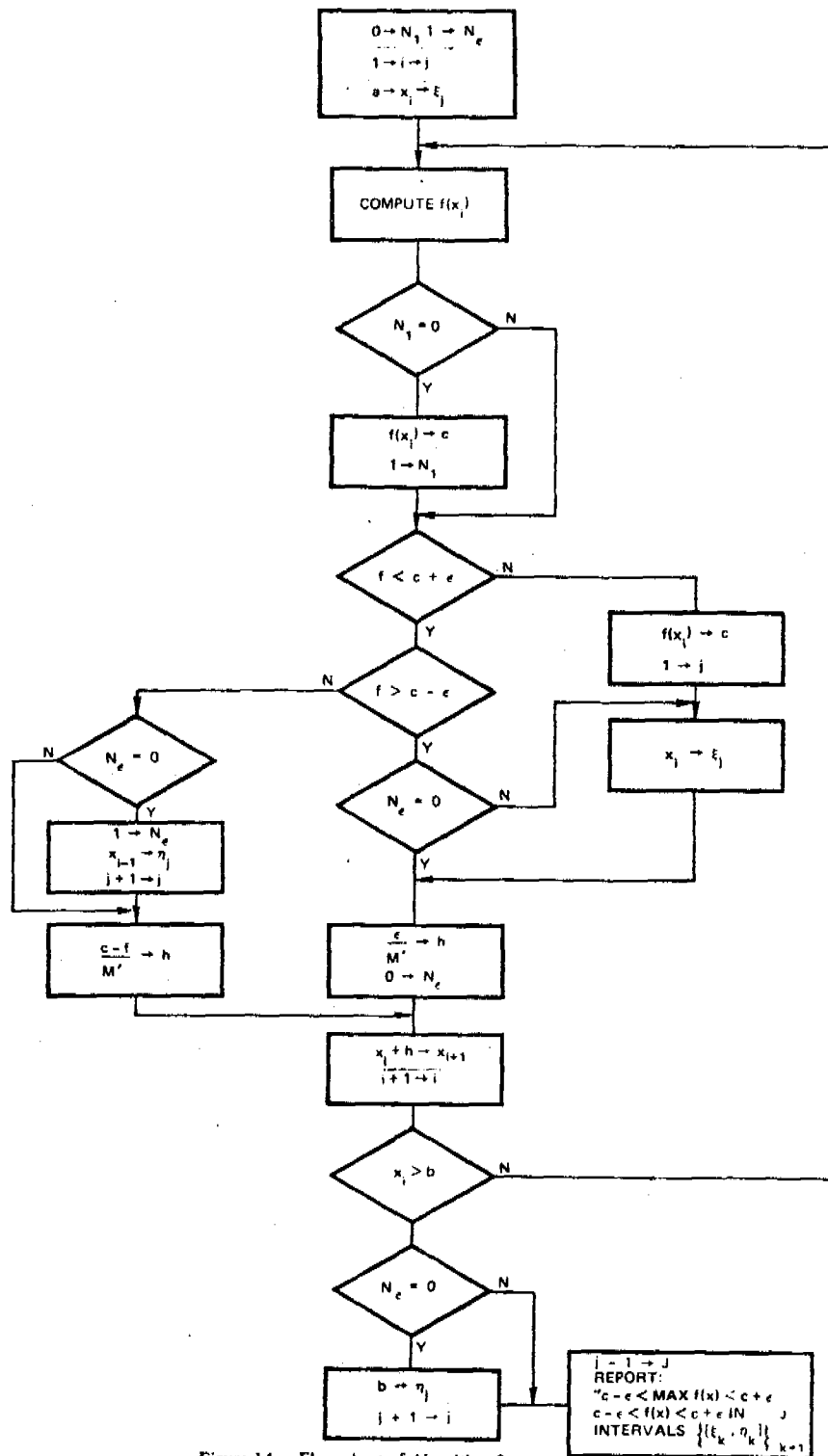


Figure 14. Flow chart of Algorithm 3



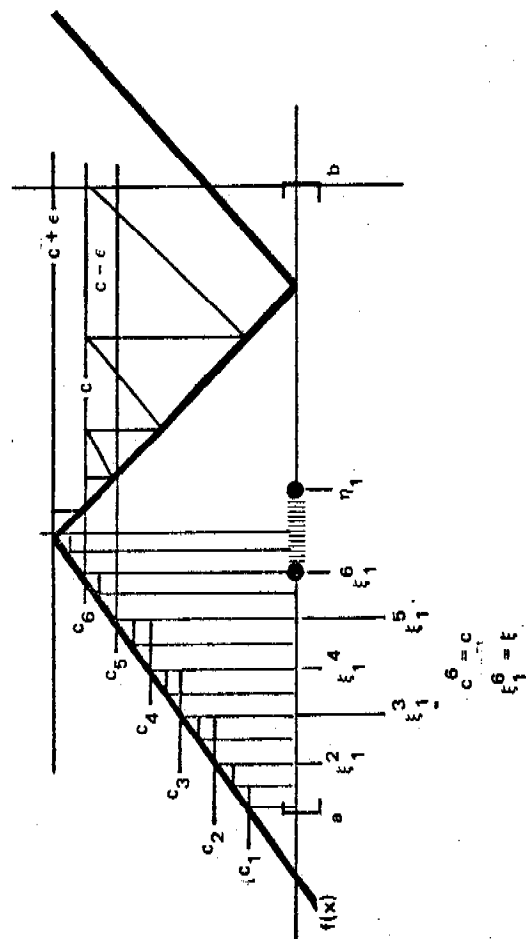


Figure 15. Search for a maximum

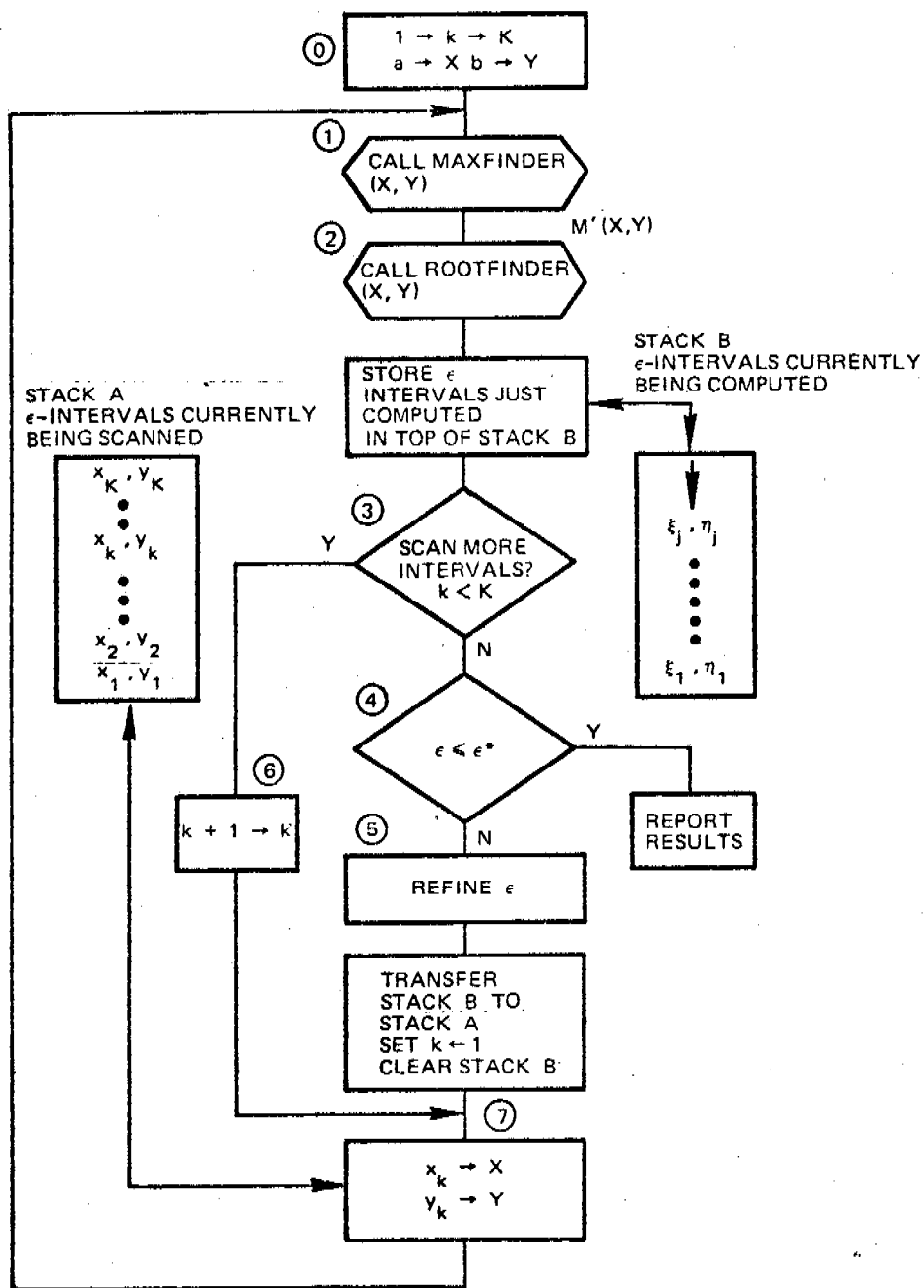


Figure 16. Flow chart of Algorithm 4: GNEWTON

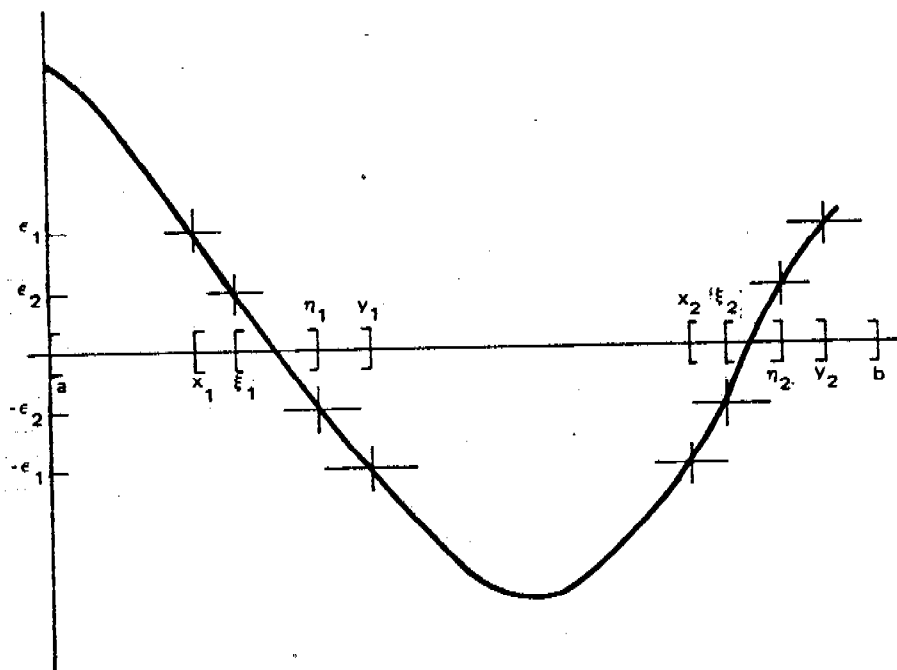


Figure 17. Rootfinding using GNEWTON

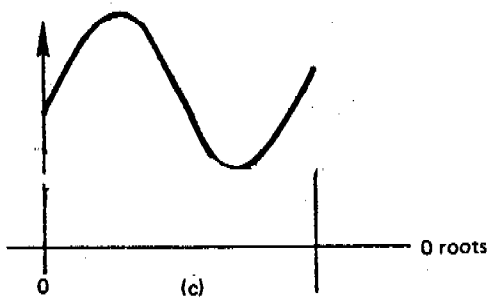
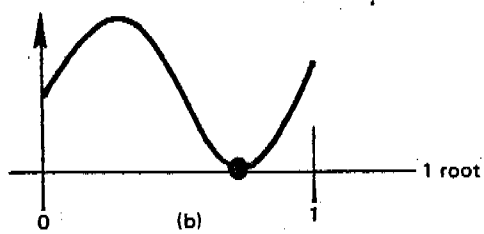
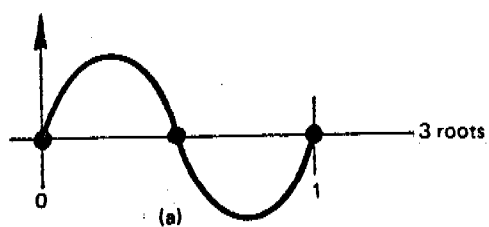


Figure 18. The function  $f_k = k + \sin 2\pi x$  on  $[0, 1]$

- (a)  $k = 0$
- (b)  $k = 1$
- (c)  $k = 2$

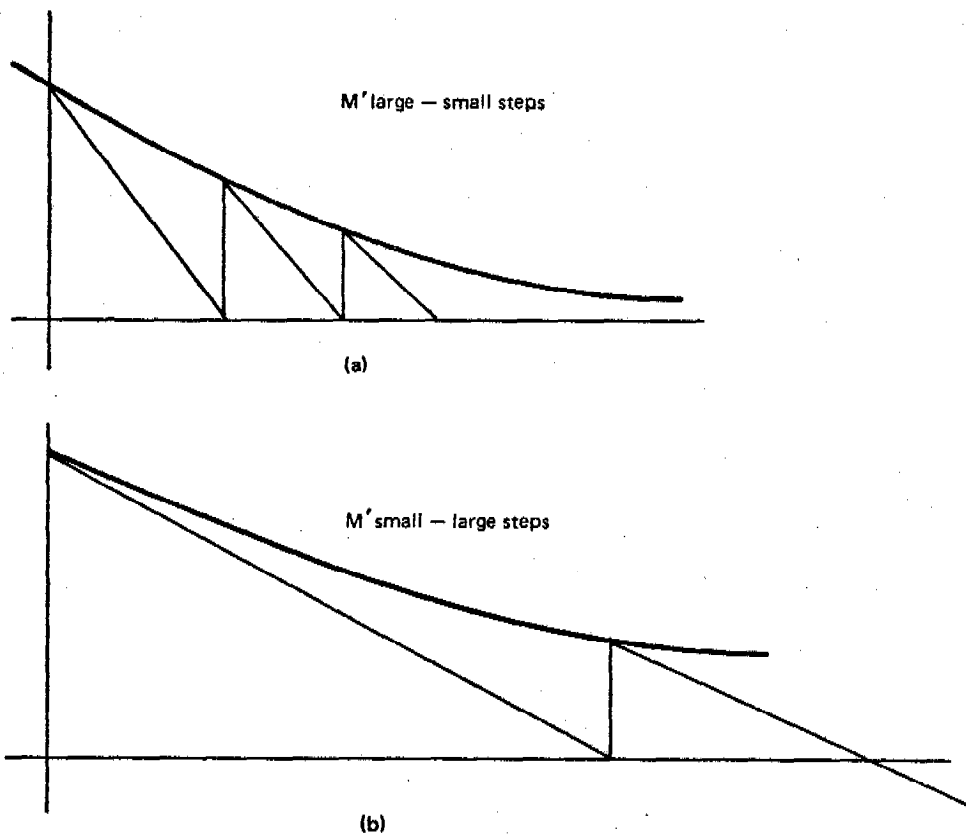


Figure 19. Effect of  $M'$  on step size of ROOTFINDER

# AN IMPROVED ITERATIVE METHOD FOR OPTIMIZING SYMMETRIC SUCCESSIVE OVERRELAXATION

Vitalius Benokraitis  
Applied Mathematics and Sciences Laboratory  
U.S. Army Ballistic Research Laboratories  
Aberdeen Proving Ground, Maryland 21005

**ABSTRACT.** An algorithm is proposed to determine the optimum relaxation parameter  $\omega_0$  as well as the spectral radius of the iteration matrix  $S_{\omega_0}$  corresponding to the symmetric successive overrelaxation method. The algorithm is based on the work of Evans and Forrington and Young. Computational results indicate that the improved algorithm converges to the optimum parameter even when the scheme of Evans and Forrington fails.

## 1. INTRODUCTION. Consider the linear system

$$(1) \quad Au = b$$

where  $A$  is a real, symmetric, positive definite matrix of order  $N$ . The real  $N$ -vector  $b$  is given and the  $N$ -vector  $u$  is to be determined.

Systems of the form (1) arise in the finite difference solution of boundary value problems involving elliptic partial differential equations. In particular, we shall be concerned with the generalized Dirichlet problem involving the differential equation

$$(2) \quad L[u] = \frac{\partial}{\partial x} \left( A \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left( C \frac{\partial u}{\partial y} \right) + Fu = T$$

where\*  $A = A(x,y) > 0$ ,  $C = C(x,y) > 0$ , and  $F = F(x,y) \leq 0$  in  $R \cup S$ . Here  $R$  is a bounded connected plane region and  $S$  is its boundary. Given a function  $g(x,y)$  continuous on  $S$ , we seek a function  $u(x,y)$  twice continuously differentiable in  $R$  and continuous on  $S$  such that  $L[u] = T$  in  $R$  and  $u = g$  on  $S$ .

In order to apply the method of finite differences, we superimpose over the region  $R$  a grid consisting of a network of horizontal and vertical lines spaced at intervals of  $h = \Delta x = \Delta y$  units apart. For simplicity, we assume the spacing to be uniform, although this is not a necessary requirement.

Now for some  $h_0$  and some  $(x_0, y_0)$  in  $R$ , we consider the set  $\Omega_{h_0}$  which contains all points of the form  $(x_0 + ih_0, y_0 + jh_0)$  for integers  $i$  and  $j$ . Following Young [1974], we assume that for any point of  $\Omega_{h_0}$  which lies in  $R$ , the four adjacent points lie in  $R$  or on  $S$ . Furthermore, this property is assumed to hold for all  $h$  such that  $h_0/h$  is an integer. Moreover, we define  $R_h = \Omega_h \cap R$  and  $S_h = \Omega_h \cap S$ .

\*The coefficient  $A = A(x,y)$  should not be confused with the matrix  $A$  of the system (1).

At any point  $(x, y)$  of  $R_h = \Omega_h \cap R$ , the differential equation given by (2) is replaced by the symmetric difference equation

$$(3) \quad L_h[u] = \frac{1}{h^2} \left\{ A\left(x + \frac{h}{2}, y\right) [u(x + h, y) - u(x, y)] \right. \\ - A\left(x - \frac{h}{2}, y\right) [u(x, y) - u(x - h, y)] \\ + C\left(x, y + \frac{h}{2}\right) [u(x, y + h) - u(x, y)] \\ \left. - C\left(x, y - \frac{h}{2}\right) [u(x, y) - u(x, y - h)] \right\} \\ + F(x, y) u(x, y) = T(x, y).$$

Thus we have transformed the continuous problem to a discrete generalized Dirichlet problem. That is, we now seek to determine a function  $u$  defined on  $R_h \cup S_h$  such that  $L_h[u] = T(x, y)$  on  $R_h$  and  $u = g$  on  $S_h$ .

Multiplying (3) by  $-h^2$  and bringing the known boundary values to the right-hand side yields a system of the form (1) where the order of the matrix  $A$ , assumed to be  $N$ , is the number of mesh points in  $R_h$ . Besides being positive definite the matrix  $A$  can be shown to have Property A. Also it can be verified that the matrix  $A$  is an L-matrix, irreducible, and weakly diagonally dominant (Young [1971]).

Another property of the matrix  $A$ , which naturally (but not exclusively) points to iterative techniques as a mode of solution for the system (1) is that  $A$  is large (the order is about  $10^3$  to  $10^6$ ) and sparse (i.e., the number of nonzero elements is small compared to the total number of elements in  $A$ ).

The iterative scheme on which we shall focus our attention is the symmetric successive overrelaxation (SSOR) method. In particular, we shall be interested in obtaining certain parameters of an accelerated SSOR method.

2. THE SSOR METHOD. In order to define the SSOR method it is convenient to rewrite (1) in the form

$$(4) \quad u = Bu + c$$

where

$$(5) \quad \begin{cases} B = I - D^{-1}A \\ C = D^{-1}b \end{cases} \quad A = L + U$$

and where  $D = \text{diag}(A)$  and  $L$  and  $U$  are strictly lower and upper triangular matrices, respectively.

The SSOR method is defined as follows (Sheldon [1955]). Let  $u^{(0)}$  be an arbitrary initial approximation to the solution of the system  $Au = b$ , and define the sequence  $u^{(1/2)}, u^{(1)}, u^{(3/2)}, u^{(2)}, \dots$ , by

$$(6) \quad \begin{aligned} u^{(n+1/2)} &= \omega(Lu^{(n+1/2)} + Uu^{(n)} + c) + (1 - \omega) u^{(n)} \\ u^{(n+1)} &= \omega(Lu^{(n+1/2)} + Uu^{(n+1)} + c) + (1 - \omega) u^{(n+1/2)} \\ n &= 0, 1, 2, \dots \end{aligned}$$

Eliminating  $u^{(n+1/2)}$  we get

$$(7) \quad u^{(n+1)} = S_{\omega} u^{(n)} + k_{\omega}$$

where

$$\begin{aligned} S_{\omega} &= U_{\omega} L_{\omega} \\ &= (I - \omega U)^{-1} (\omega L + (1 - \omega) I) (I - \omega L)^{-1} (\omega U + (1 - \omega) I) \\ k_{\omega} &= \omega(2 - \omega) (I - \omega U)^{-1} (I - \omega L)^{-1} c \end{aligned}$$

One of our goals will be to determine the relaxation parameter so that the rate of convergence of our method is optimized in some sense.

Let us look at the SSOR method a little closer. Anyone familiar with SOR, will recognize that each SSOR iteration is composed of two SOR half iterations. The first half iteration is just a "normal" SOR iteration. The second half iteration is another SOR iteration but taking the equations in reverse order.

Even though one SSOR iteration is composed of two SOR iterations, there is a way to reduce the work required for each SSOR iteration by providing storage space for an extra  $N$ -vector. How this is accomplished can be seen by explicitly exhibiting two full SSOR iterations and noting that certain vectors are repeated in a pair of half-iterations.

$$\left\{ \begin{aligned} u^{(n+1/2)} &= \omega(Lu^{(n+1/2)} + Uu^{(n)} + c) + (1 - \omega) u^{(n)} \\ &\quad \text{save } Lu^{(n+1/2)} \\ u^{(n+1)} &= \omega(Lu^{(n+1/2)} + Uu^{(n+1)} + c) + (1 - \omega) u^{(n+1/2)} \\ &\quad \text{save } Uu^{(n+1)} \\ u^{(n+3/2)} &= \omega(Lu^{(n+3/2)} + Uu^{(n+1)} + c) + (1 - \omega) u^{(n+1)} \\ &\quad \text{save } Lu^{(n+3/2)} \\ u^{(n+2)} &= \omega(Lu^{(n+3/2)} + Uu^{(n+2)} + c) + (1 - \omega) u^{(n+3/2)} \\ &\quad \text{save } Uu^{(n+2)} \\ &\vdots \end{aligned} \right.$$

This scheme is due to Niethammer [1964]. The work required per iteration using this technique is approximately the same as with the SOR method.

We noted that the SSOR method can be written as

$$u^{(n+1)} = S_{\omega} u^{(n)} + k_{\omega}$$

with  $S_{\omega}$  and  $k_{\omega}$  appropriately defined.

Evidently, SSOR is a stationary iterative method of first degree. It is stationary because  $S_{\omega}$  and  $k_{\omega}$  are fixed from iteration to iteration. It is of first degree since  $u^{(n+1)}$  depends only on one preceding iterate  $u^{(n)}$ . It is well known (Young [1971]) that we must have  $S(S_{\omega}) < 1$  in order to guarantee convergence. Here  $S(S_{\omega})$  denotes the spectral radius of  $S_{\omega}$ . This condition holds if  $0 < \omega < 2$  and  $A$  is positive definite. Also, if we define the error

$$e^{(n)} = u^{(n)} - \bar{u}$$

where  $\bar{u}$  is the exact solution of  $Au = b$ , we have approximately (in some appropriate norm)

$$||e^{(n)}|| \approx S(S_{\omega}) ||e^{(n-1)}||$$

and

$$||e^{(n)}|| \approx [S(S_{\omega})]^n ||e^{(0)}||$$

To reduce  $||e^{(n)}||$  to a fraction, say  $\zeta$ , of  $||e^{(0)}||$  we must have

$$\frac{||e^{(n)}||}{||e^{(0)}||} \approx [S(S_{\omega})]^n < \zeta$$

and we must iterate  $n$  times, where

$$n \doteq \frac{-\log \zeta}{-\log S(S_{\omega})}$$

We define  $R(S_{\omega}) = -\log S(S_{\omega})$  as the rate of convergence of the SSOR method.

As indicated, we suspect that the rate of convergence of the SSOR method

$$R(S_{\omega}) = -\log S(S_{\omega})$$

depends on  $\omega$ . Not evident from our notation, the rate of convergence is also governed by the ordering of the equations. Given an ordering, the optimum  $\omega$  will be considered to be the  $\omega$  which minimizes  $S(S_{\omega})$  or maximizes  $R(S_{\omega})$ . Assuming the "natural ordering", for a certain "good" choice of  $\omega = \omega_1$ , which depends on upper bounds of  $S(B)$  and  $S(LU)$ , Young [1974] has shown that the



number of iterations necessary to solve the discrete generalized Dirichlet problem is proportional to  $h^{-1}$ . This is the same order of magnitude required by the SOR method with optimum  $\omega$ .

Thus, even if Niethammer's work-saving techniques are employed, there seems to be little justification to choose SSOR over SOR.

However, what is nice about the SSOR method is that the eigenvalues of the iteration matrix  $S_\omega$  are real and nonnegative (Young, [1971]). Under these conditions, it is possible to accelerate SSOR by an order of magnitude by means of semi-iteration. (This is not possible for the SOR method, since many of the eigenvalues of its iteration matrix  $L_\omega$  are complex for the optimal  $\omega$ .)

3. SSOR SEMI-ITERATION. Semi-iteration was studied by Varga (1957) and Golub and Varga (1961). The optimum semi-iterative method based on SSOR, denoted by SSOR-SI, is defined by

$$u^{(n+1)} = \rho_{n+1} \{ \bar{\rho} (S_\omega u^{(n)} + k_\omega) + (1 - \bar{\rho}) u^{(n)} \} + (1 - \rho_{n+1}) u^{(n-1)}$$

Here

$$\bar{\rho} = \frac{2}{2 - S(S_\omega)}$$

$$\rho_1 = 1$$

$$\rho_2 = (1 - \sigma^2/2)^{-1}$$

$$\rho_{n+1} = \left(1 - \frac{\sigma^2 \rho_n}{4}\right)^{-1}, \quad n = 2, 3, \dots$$

where

$$\sigma = \frac{S(S_\omega)}{2 - S(S_\omega)}$$

We see that in order to apply SSOR-SI, we must estimate the parameter  $S(S_\omega)$  along with  $\omega$ . Using estimates derived, again, by Young [1974], the required number of iterations is proportional to  $h^{-1/2}$ , an order of magnitude better than SOR.

4. OPTIMAL PARAMETERS. The parameters  $\omega$  and  $S(S_\omega)$  determined by Young are not optimal. Habetler and Wachspress (1961) determined what they presumed were the optimum parameters:

$$\omega_0 = \frac{2}{1 + \sqrt{1 - 2\alpha + 4\beta}}$$

$$S(S_{\omega_0}) = \frac{1 - \frac{1 - \alpha}{\sqrt{1 - 2\alpha + 4\beta}}}{1 + \frac{1 - \alpha}{\sqrt{1 - 2\alpha + 4\beta}}}$$

where

$$\alpha = \frac{(v, DBv)}{(v, Dv)}$$

$$\beta = \frac{(v, DLUv)}{(v, Dv)}$$

and where  $v$  is a vector such that  $S_{\omega_0} v = S(S_{\omega_0})v$ . However, these formulas are not usable directly, since these are implicit relationships. Thus we are unable to determine  $\omega_0$  and  $S(S_{\omega_0})$  from the formulas since the formulas imply that we know the eigenvector  $v$  (and therefore  $S(S_{\omega_0})$  and  $\omega_0$ ).

Evans and Forrington [1963] used an iterative technique based on these formulas to determine the optimum  $\omega$  and corresponding  $S(S_{\omega})$ . Starting with an initial guess of  $\omega$ , the power method on the matrix  $S_{\omega}$  was used to determine an approximation to the eigenvector associated with  $S(S_{\omega})$ . Then, via  $\alpha$  and  $\beta$ ,  $\omega$  and  $S(S_{\omega})$  were updated by the formulas, and the procedure was repeated until  $\omega$  and  $S(S_{\omega})$  settled down.

This procedure worked quite well for Laplace's equation. However, our numerical results indicate that for certain cases where  $S(LU) < \frac{1}{4}$ , the Habetler-Wachspress formulas do not hold, and in turn, the Evans-Forrington procedure fails. It was found that the optimum parameters seem to be

$$\omega_0 = \frac{2}{1 + \sqrt{1 - 4\beta}}$$

$$S(S_{\omega_0}) = \frac{1 - \sqrt{1 - 4\beta}}{1 + \sqrt{1 - 4\beta}}$$

Incorporating these formulas into the Evans-Forrington scheme we have a new method for determining the optimum parameters  $\omega_0$  and  $S(S_{\omega_0})$  for the SSOR-SI method. The algorithm follows

1. Choose convergence tolerances  $\epsilon_1$  and  $\epsilon_2$  and initial values of  $\omega$  and  $v \neq 0$ .
2. Iterate with the power method to obtain  $S(S_{\omega})$  and a vector  $v$  such that
 
$$S_{\omega} v = S(S_{\omega})v$$

3. Compute

$$\alpha = (v, DBv)/(v, Dv)$$

$$\beta = (v, DLUv)/(v, Dv)$$

4. Compute

$$\omega' = \begin{cases} \frac{2}{1 + \sqrt{1 - 2\alpha + 4\beta}} & \text{if } \alpha \leq 4\beta \\ \frac{2}{1 + \sqrt{1 - 4\beta}} & \text{if } \alpha > 4\beta \end{cases}$$

$$S' = \begin{cases} \left(1 - \frac{1 - \alpha}{\sqrt{1 - 2\alpha + 4\beta}}\right) / \left(1 + \frac{1 - \alpha}{\sqrt{1 - 2\alpha + 4\beta}}\right) & \text{if } \alpha \leq 4\beta \\ \frac{1 - \sqrt{1 - 4\beta}}{1 + \sqrt{1 - 4\beta}} & \text{if } \alpha > 4\beta \end{cases}$$

5. Terminate process if

$$|\omega - \omega'| < \epsilon_1$$

$$|S(S_\omega) - S'| < \epsilon_2$$

and choose

$$\omega_0 = \omega'$$

$$S(S_{\omega_0}) = S'$$

Otherwise set  $\omega = \omega'$  and go to Step 2.

**5. NUMERICAL EXAMPLE.** We now apply the algorithm to a specific discrete generalized Dirichlet problem of the form (3). In particular, let

$$A(x,y) = C(x,y) = e^{10(x+y)}$$

$$F(x,y) = T(x,y) = 0$$

and let the region be the unit square with zero boundary values except unity on side  $y = 0$ . Initially, a "good"  $\omega$  (Young [1974]) was chosen. The tolerances  $\epsilon_1 = \epsilon_2 = 10^{-4}$  were specified. Results are given in Table 1.

TABLE 1  
OPTIMUM PARAMETERS  $\omega_0$  AND  $S(S_{\omega_0})$  OBTAINED BY ALGORITHM

$h^{-1}$	$\omega_0$	$S(S_{\omega_0})$	Number of Iterations	$S(LU)$
20	1.5866	.5866	5	.2329
40	1.7653	.7653	15	.2455
80	1.8742	.8742	35	.2489

The Evans-Forrington algorithms did not converge in this case. For a more detailed discussion of this example and a number of other problems, see Benokraitis [1974].

6. ACKNOWLEDGEMENT. This is a small part of a thesis written under the direction of Professor David M. Young, whose guidance is gratefully acknowledged.

#### 7. REFERENCES.

- Benokraitis, V.J. [1974] "On the Adaptive Acceleration of Symmetric Successive Overrelaxation", Doctoral Thesis, University of Texas, Austin.
- Evans, D.J. and C.V.D. Forrington [1963], "An Iterative Process for Optimizing Symmetric Successive Overrelaxation", Computer J., 6, 271-273.
- Golub, G.H. and R. S. Varga [1961], "Chebyshev Semi-Iterative Methods, Successive Overrelaxation Iterative Methods, and Second-Order Richardson Iterative Methods", Numer. Math., Parts I and II, 3, 147-168.
- Habetler, G.J., and E.L. Wachspress [1961], "Symmetric Successive Overrelaxation in Solving Diffusion Difference Equations", Math. Comp. 15, 356-362.
- Niethammer, W. [1964], "Relaxation bei Komplexen Matrizen", Math. Zeitsch 86, 34-40.
- Sheldon, J.W. [1955], "On the Numerical Solution of Elliptic Difference Equations", Math. Tables Aids Comput. 9, 101-112.
- Varga, R.S. [1957], "A Comparison of the Successive Overrelaxation Method and Semi-Iterative Methods Using Chebyshev Polynomials", J. Soc. Indus. Appl. Math. 5, 39-46.
- Young, D.M. [1971], Iterative Solution of Large Linear Systems, Academic Press, New York.
- Young, D.M. [1974], "On the Accelerated SSOR Method for Solving Large Linear Systems", Report CNA-92, Center for Numerical Analysis, University of Texas, Austin.

Royce W. Soanes Jr.  
Watervliet Arsenal, Watervliet, New York

1. Introduction. Since the cubic spline is a special case of the Variable Power - splines presented here, the latter may be regarded as a generalization of the former. This generalization does not take place in the direction of obtaining higher degrees of differentiability, however, but rather in the more practical direction of obtaining greater flexibility and better local behavior while retaining two orders of differentiability. The cubic spline has the optimal property of being the interpolater of smallest quadratic mean second derivative; this is a global property, however, and as such may conflict with the curve fitter's desire for certain local behavior. Specifically, the cubic spline has a tendency to misrepresent the behavior of a function which passes from a region of low curvature through a region of high curvature. The cubic may underestimate the high curvature and overestimate the low. This tendency may be manifested in the cubic interpolater by the presence of inflection points where none are desired. VP - splines make it possible to develop a more aesthetically pleasing functional curve from a given set of data by eliminating or at least diminishing undesirable local behavior when it is encountered.

# NOMENCLATURE

$x_1 < x_2 < \dots < x_N$  is a sequence of nodes over an interval on which we desire to interpolate the function  $y$ .

$l_i = x_{i+1} - x_i$  = length of the  $i$ th subinterval.

$y_i$  = known function value at the  $i$ th node.

$y_i', y_i''$  = unknown first and second derivatives at the  $i$ th node.

$y_i(x), y_i'(x), y_i''(x)$  = interpolating function, derivative and second derivative at any point in the  $i$ th subinterval.

$r_i = (x - x_i) / l_i$

$q_i = (y_{i+1} - y_i) / l_i$

2. The Basic Form. Consider the following interpolatory function defined on the  $i$ th subinterval.

$$(1) \quad k_i y_i(x) = a_i + b_i r_i + c_i r_i^{m_i} + d_i (1 - r_i)^{n_i}$$

The parameters  $m_i$  and  $n_i$  are positive real numbers and  $k_i = m_i + n_i - m_i n_i$ . To insure a bounded second derivative,  $m_i$  and  $n_i$  should both be greater than 2. The functions  $y_i(x)$  and  $y_i'(x)$  can be evaluated at  $x_i$  and  $x_{i+1}$  to obtain  $a_i, b_i, c_i$  and  $d_i$  in terms of  $y_i, y_{i+1}, y_i'$  and  $y_{i+1}'$ .

$$(2) \quad a_i = k_i y_i + l_i (m_i q_i - (m_i - 1) y_i' - y_{i+1}')$$

$$(3) \quad b_i = \ell_i (-m_i n_i q_i + m_i y_i' + n_i y_{i+1}') \quad$$

$$(4) \quad c_i = \ell_i (n_i q_i - y_i' - (n_i - 1) y_{i+1}') \quad$$

$$(5) \quad d_i = \ell_i (-m_i q_i + (m_i - 1) y_i' + y_{i+1}') \quad$$

The interpolater is only once differentiable at this juncture, so we enforce continuity of its second derivative at the  $i$ th node by setting  $y_i''(x_i) = y_{i-1}''(x_i)$  and obtaining Eq. (6).

$$(6) \quad [\ell_i m_{i-1} k_i (1 - k_{i-1}) + \ell_{i-1} n_i k_{i-1} (1 - k_i)] y_i' \\ = \ell_i k_i m_{i-1} (m_{i-1} - 1) (n_{i-1} q_{i-1} - y_{i-1}') + \\ \ell_{i-1} k_{i-1} n_i (n_i - 1) (m_i q_i - y_{i+1}') \quad$$

Once  $m_i$  and  $n_i$  are set for each subinterval, we may solve the linear system represented by Eq. (6) (plus end conditions) and obtain the nodal derivatives which insure the continuity of the second derivative of the interpolater. Setting  $m_i = n_i = 3$  on each subinterval would give us the cubic spline formula.

3. Effect of Large  $m$ 's and  $n$ 's. Some insight into the effect of increasing the  $m$ 's and  $n$ 's may be obtained from the consideration of a special case. Equation (7) may be obtained from Eq. (6) by setting  $m_i = n_i = m$  on each subinterval.

$$(7) \quad (m-1)(\ell_i + \ell_{i-1}) y_i' = \ell_i (m q_{i-1} - y_{i-1}') + \ell_{i-1} (m q_i - y_{i+1}') \quad$$

It is apparent from Eq. (7) that the corresponding linear system will become increasingly more diagonally dominant as  $m$  becomes large, and in the limit as  $m$  approaches infinity,  $y_i'$  will be given by Eq. (8).

$$(8) \quad y_i' = (\ell_i q_{i-1} + \ell_{i-1} q_i) / (\ell_i + \ell_{i-1}).$$

At the same time, Equations (1) through (5) show that  $y_i(x)$  approaches  $y_i + r_i \ell_i q_i$  (the linear interpolater) as  $m$  approaches infinity. This example indicates that we should obtain a well conditioned set of linear equations as we judiciously increase the  $m$ 's and  $n$ 's to obtain the flattening effect which will produce desirable local behavior in the VP interpolater.

4. Elimination of Improper Inflection Points. When the cubic spline is used for interpolation, the curve fitter may be faced with the undesirable situation of having  $y_i'' y_{i+1}'' < 0$  while  $(q_i - y_i')(q_i - y_{i+1}') < 0$ . These conditions indicate the presence of an inflection point in the interior of the  $i$ th subinterval when  $q_i$  is between  $y_i'$  and  $y_{i+1}'$ . This is an aesthetically displeasing situation created by the average curvature diminishing property of the cubic spline. The cubic is trying to deny the existence of a point of high curvature in the function. In order to eliminate the undesirable inflection point, of which there can be at most one per subinterval, we must insure that  $y_i'' y_{i+1}'' \geq 0$ . This is equivalent to having  $c_i d_i \geq 0$ . We therefore want to enforce condition (9) in the event that  $q_i$  is between



$y'_i$  and  $y'_{i+1}$ .

$$(9) \quad [n_i(q_i - y'_{i+1}) - y'_i + y'_{i+1}][m_i(y'_i - q_i) - y'_i + y'_{i+1}] \geq 0$$

But since  $(q_i - y'_i)(q_i - y'_{i+1}) < 0$ , condition (9) can obviously be enforced by insuring that  $m_i$  and  $n_i$  obey conditions (10) and (11).

$$(10) \quad m_i \geq (y'_{i+1} - y'_i) / (q_i - y'_i)$$

$$(11) \quad n_i \geq (y'_{i+1} - y'_i) / (y'_{i+1} - q_i)$$

Conditions (10) and (11) must naturally be enforced in an iterative manner if two orders of differentiability are desired, since  $m_i$  and  $n_i$  must be set before  $y'_i$  and  $y'_{i+1}$  are obtained.

5. A Local Consideration. A local VP - spline over the restricted node set  $[x_{i-1}, x_i, x_{i+1}]$  will be found useful in the initial setting of the  $m$ 's and  $n$ 's. If we set  $y''_i(x_{i+1}) = 0 = y''_{i-1}(x_{i-1})$ , we obtain the following end conditions.

$$(12) \quad (n_i - 1)y'_{i+1} = n_i q_i - y'_i$$

$$(13) \quad (m_{i-1} - 1)y'_{i-1} = m_{i-1} q_{i-1} - y'_i$$

Equations (12) and (13) can then be used to eliminate  $y'_{i+1}$  and  $y'_{i-1}$  from Eq. (6) to yield Eq. (14).

$$(14) \quad y'_i = (\ell_i m_{i-1} q_{i-1} + \ell_{i-1} n_i q_i) / (\ell_i m_{i-1} + \ell_{i-1} n_i)$$

Equation (14) can now be solved for the ratio of  $n_i$  to  $m_{i-1}$ .

$$(15) \quad R_i = n_i/m_{i-1} = (\ell_i/\ell_{i-1})(q_{i-1}-y'_i)/(y'_i-q_i)$$

If we now set  $y'_i$  at each node according to some preferred local formula, we may set the m's and n's according to the following rule.

$$(16) \quad m_{i-1} = L \text{ and } n_i = LR_i \text{ if } R_i > 1$$

$$n_i = L \text{ and } m_{i-1} = L/R_i \text{ if } R_i < 1$$

Rule (16) sets a lower bound of L on the m's and n's and assumes that  $R_i$  is positive and finite. If Eq. (15) and Rule (16), are used to initialize the m's and n's, Conditions (10) and (11) can be used in conjunction with Eq. (6) to iteratively eliminate any unwanted inflection points.

A simple local formula which has been found effective in practice for initial setting of the nodal derivatives is given by Eq. (17).

$$(17) \quad y'_i = (1-w_i) q_{i-1} + w_i q_i$$

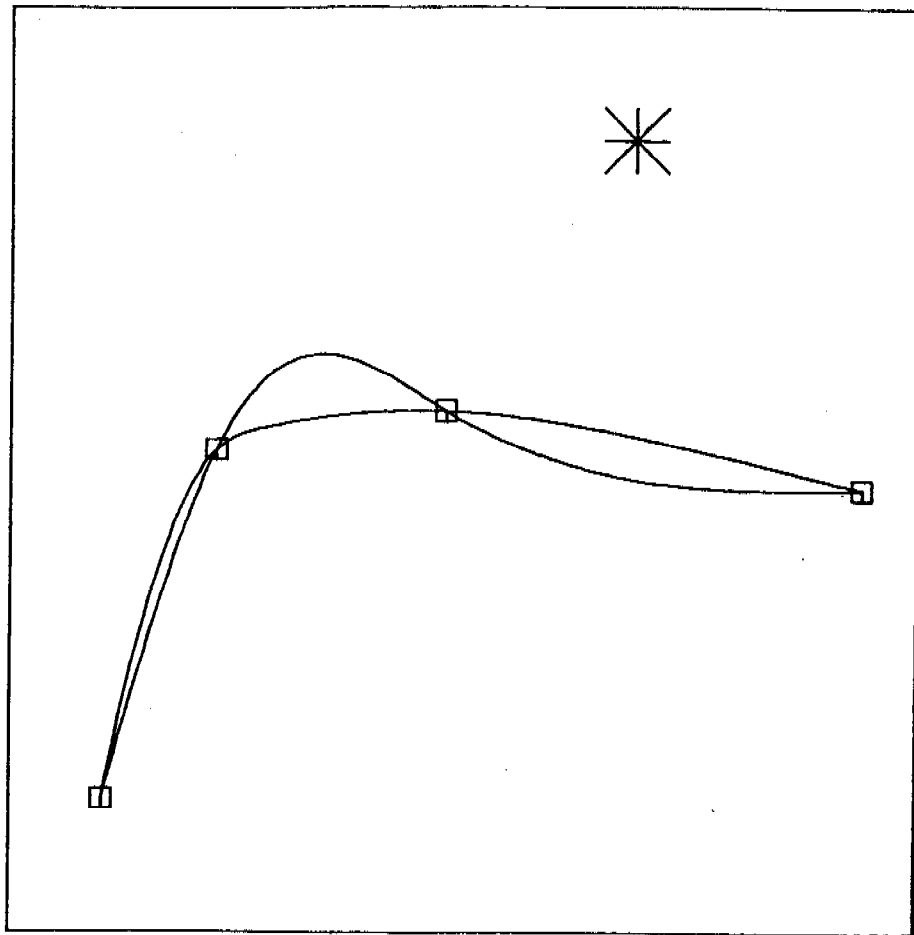
$$\text{where } w_i = 1/\{1 + \sqrt{[(1 + q_i^2)/(1 + q_{i-1}^2)]}\}$$

This derivative is obtained from the slope of the line through  $(x_i, y_i)$  which makes equal angles with the left and right chords. When this formula is used, it is usually unnecessary to increase the m's and n's gradually in order to enforce conditions (10) and (11).

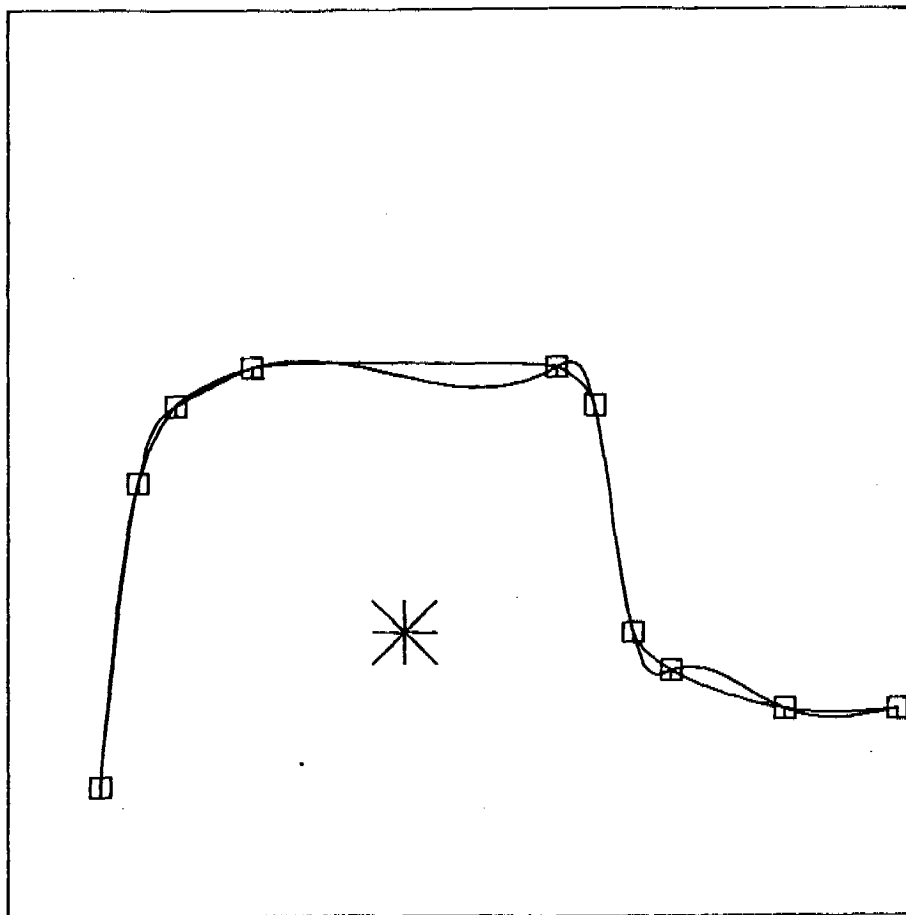
6. A Simple Computational Procedure. An abbreviated procedure which can be followed in most cases is:

- I. Set  $L$  equal to a value between 2 and 3. Values greater than 3 may flatten the curve too much between the nodes. Values closer to 2 will produce more roundedness in the interpolater.
- II. Compute the initial nodal derivatives according to Eq. (17) for  $2 \leq i \leq N - 1$ .
- III. Set  $n_1 = L = m_{N-1}$  and calculate  $R_i$  for  $2 \leq i \leq N - 1$  using Eq. (15).
- IV. Compute the rest of the  $m$ 's and  $n$ 's using Rule (16).
- V. Solve the tridiagonal set of equations represented by Equations (6), (12) and (13) for the final nodal derivatives.

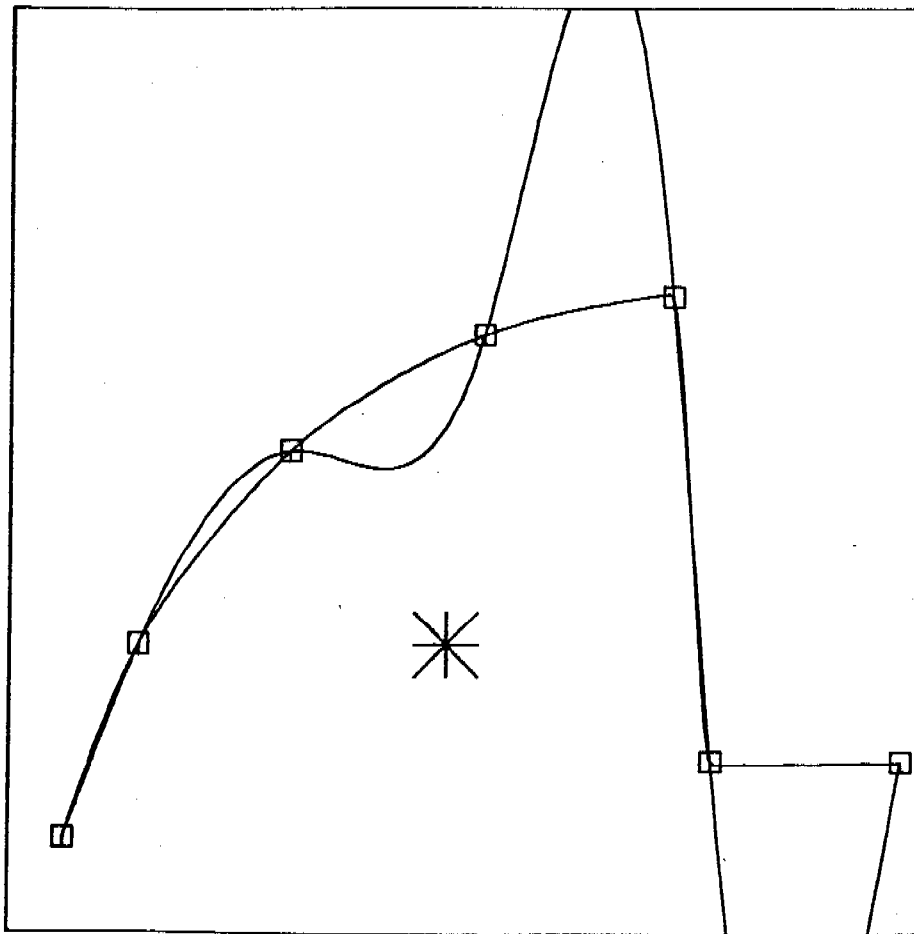
7. Examples. The following drawings, which were produced on a graphic display CRT, illustrate the more stable behavior of the VP-spline as compared with the oscillatory behavior of the cubic spline. Each of the VP-spline curves was computed using the abbreviated procedure previously outlined [not making use of conditions (10) and (11)].



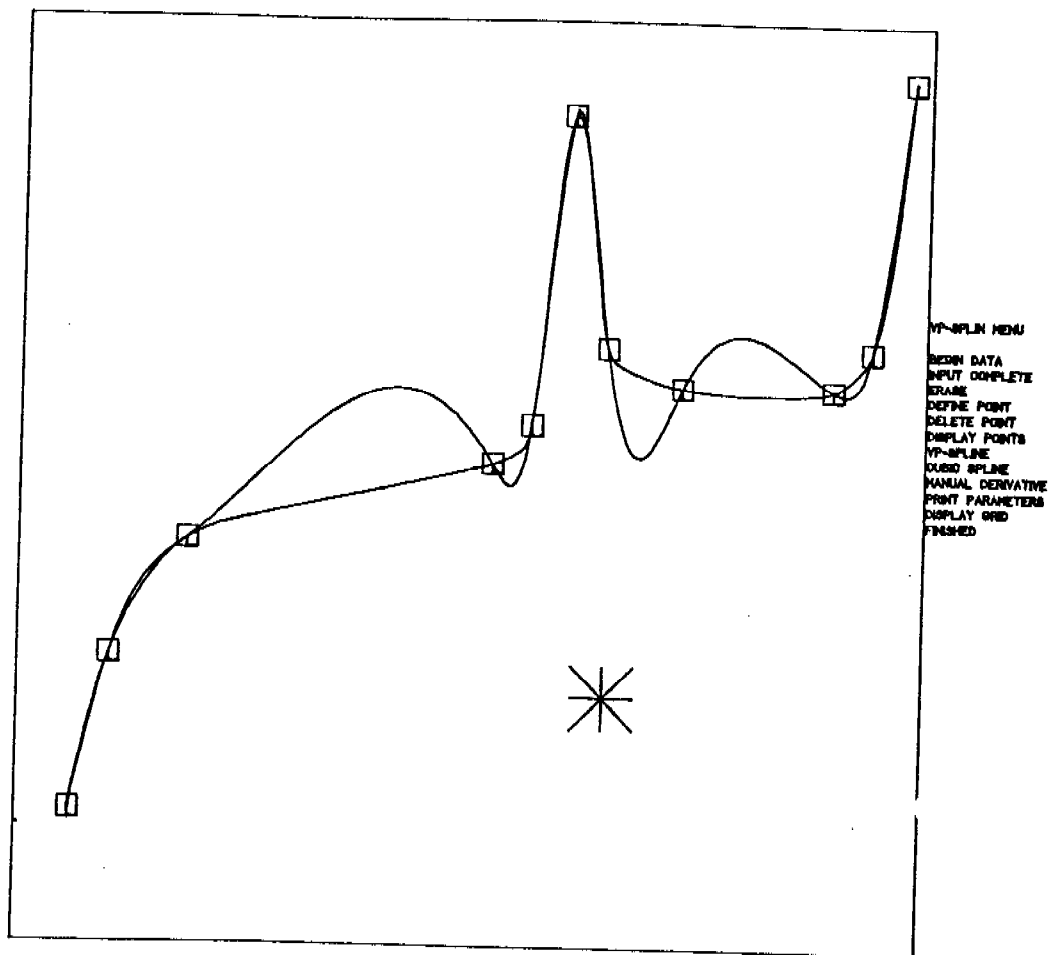
VF-SPLIN MENU  
 BEGIN DATA  
 INPUT COMPLETE  
 ERASE  
 DEFINE POINT  
 DELETE POINT  
 DISPLAY POINTS  
 VF-SPLINE  
 CURVE SPLINE  
 MANUAL DERIVATIVE  
 PRINT PARAMETERS  
 DISPLAY GRID  
 FINISH



VP-SPLINE MENU  
BEGIN DATA  
INPUT COMPLETE  
ERASE  
DEFINE POINT  
DELETE POINT  
DISPLAY POINTS  
VP-SPLINE  
CURVE SPLINE  
MANUAL DERIVATIVE  
PRINT PARAMETERS  
DISPLAY GRID  
FINISHED



VP-SPLIN MENU  
 READ DATA  
 INPUT COMPLETE  
 ERASE  
 DEFINE POINT  
 DELETE POINT  
 DISPLAY POINTS  
 VP-SPLINE  
 GUIDO SPLINE  
 MANUAL DERIVATIVE  
 PRINT PARAMETERS  
 DISPLAY GRID  
 FINISHED



## REFERENCES

1. Ahlberg, J. H., Nilson, E. N., and Walsh, J. L., 1967, The Theory of Splines and Their Applications, Academic Press, New York.
2. Barnhill, R. E., and Riesenfeld, R. F., 1974, Computer Aided Geometric Design, Academic Press, New York.
3. Cline, A. K., April 1974, "Scalar- and Planar-Valued Curve Fitting Using Splines Under Tension," Comm. ACM, Vol. 17, No. 4, pp. 218-220.
4. Prenter, P. M., 1975, Splines and Variational Methods, John Wiley & Sons, New York.
5. Schweikert, D. G., 1966, "An Interpolation Curve Using a Spline in Tension," J. Math. and Physics, Vol. 45, pp. 312-317.



# ITERATIVE SOLUTION OF THE TRANSONIC POTENTIAL EQUATION

D. C. Adams and Gary Vander Roest  
U.S. Army Air Mobility R&D Laboratory  
Ames Directorate  
Moffett Field, California 94035

**ABSTRACT.** The unsteady, transonic small-disturbance equation is a suitable model for the flow on advancing rotor tips. This paper discusses the relative efficiency of two implicit iterative techniques used in the solution of this nonlinear equation; namely, the point SOR technique and the Douglas-Gunn ADI technique. While the two-dimensional calculations of both methods demonstrate the effect of varying lift and incident Mach number on a high-speed helicopter rotor, the ADI technique proved to be the more efficient method.

**1. INTRODUCTION.** The advancing blade of a helicopter in high-speed forward flight often enters the transonic flow regime. This type of flow is marked by the presence of local pockets of supersonic flow which are usually terminated by a shock. A feature of transonic flow which is beginning to be considered is that these flows are intrinsically unsteady. This is especially important for the helicopter in forward flight, since the rotor sees a constantly varying incident Mach number and angle of attack.

The first treatment of this problem was the paper of Caradonna and Isom (ref. 1) where nonlifting three-dimensional flows were considered. It was shown that the Mach number variation alone is a considerable source of unsteadiness. This treatment used a point SOR scheme to solve the set of nonlinear difference equations derived from a mixed differencing of the small-disturbance potential equation. However, SOR is often not the fastest technique at our disposal. In this paper, an iterative ADI scheme to solve the nonlinear system of difference equations is devised and then compared with the SOR method.

**2. STATEMENT OF PROBLEM.** The second-order nonlinear partial differential equation to be solved is

$$A\phi_{tt} + B\phi_{xt} = C\phi_{xx} + \phi_{yy} \quad (1)$$

where

$$\begin{aligned} A &= M_R^2 / (AR^2 \delta^{2/3}) \\ B &= [2M_R^2 / (AR \delta^{2/3})] (1 + \mu \sin \psi) \\ C &= \left[ \frac{1 - M_R^2 (1 + \mu \sin \psi)^2}{\delta^{2/3}} - (1 + \gamma) M_R^2 (1 + \mu \sin \psi) \phi_x \right] \end{aligned}$$

$\phi$  = disturbance potential

$t$  = time

$\bar{x}$  = chordwise coordinate

$\bar{y}$  = normal coordinate

$x = \bar{x}/c$

$y = (\bar{y}/c)\delta^{1/3}$

and where

AR = aspect ratio =  $r/c$

$c$  = blade chord

$M_R$  = blade incident Mach number due to rotation

$r$  = blade radius

$\bar{x}, \bar{y}$  = physical coordinates

$\gamma$  = specific heat ratio

$\delta$  = thickness ratio

$\mu$  = ratio of forward flight speed to rotational speed,  $V_\infty/\omega r$

$\psi$  = blade azimuth angle (measured from the downwind direction)

The linearized body boundary conditions are given by

$$\phi_y(x, 0^\pm) = \frac{d}{dx} [f_{u,\ell}(x, y)] \quad (2)$$

where  $f_{u,\ell}(x, y)$ , the equation for the upper and lower airfoil surface, is given by

$$f_{u,\ell} = \frac{1}{\delta} \left[ T_{u,\ell}(x) \pm \frac{\alpha}{\delta} \right] \quad (3)$$

where  $T_{u,\ell}$  is the thickness distribution and  $\alpha$  is the blade angle of attack.

The far-field boundary conditions are given by

$$\phi(x, y) = -\frac{C_L}{4\pi\delta^{2/3}} \tan^{-1} \left( \frac{\sqrt{1 - M_\infty^2}}{\delta^{1/3}} \frac{y}{x} \right), \quad x^2 + y^2 \rightarrow \infty \quad (4)$$

where

$$C_L = \frac{\text{lift per unit length}}{[(1/2)\rho v^2]c}$$

and where

$v$  = total velocity of the rotor

$\rho$  = air density

The right-hand side of eq. (1) is of mixed type, being elliptic when flow is subsonic ( $M_L < 1$ ) and hyperbolic when flow is supersonic ( $M_L > 1$ ).

Equation (1) is differenced as follows:

$$A \delta_{tt}^{\phi^{M+1}} + B \delta_{xt}^{\phi^{M+1}} = \epsilon_i C_i^M \delta_{xx}^{\phi^{M+1}} + (1 - \epsilon_{i-1}) C_{i-1}^M \bar{\delta}_{xx}^{\phi^{M+1}} + \delta_{yy}^{\phi^{M+1}} \quad (5)$$

where

$N$  = time level

$M$  = iteration level

$$\delta_{tt}^{\phi^{M+1}} = \frac{1}{\Delta t^2} \left( \phi_i^{N+1} - 2\phi_i^N + \phi_i^{N-1} \right)_{i,j}^{M+1}$$

$$\delta_{xt}^{\phi^{M+1}} = \frac{1}{\Delta x \Delta t} \left( \phi_i^{N+1} - \phi_i^N - \phi_{i-1}^{N+1} + \phi_{i-1}^N \right)_j^{M+1}$$

$$\delta_{xx}^{\phi^{M+1}} = \frac{1}{\Delta x^2} \left( \phi_{i+1}^{N+1} - 2\phi_i^{N+1} + \phi_{i-1}^{N+1} \right)_j^{M+1}$$

$$\bar{\delta}_{xx}^{\phi^{M+1}} = \frac{1}{\Delta x^2} \left( \phi_i^{N+1} - 2\phi_{i-1}^{N+1} + \phi_{i-2}^{N+1} \right)_j^{M+1}$$

$$\delta_{yy}^{\phi^{M+1}} = \frac{1}{\Delta y^2} \left( \phi_{j+1}^{N+1} - 2\phi_j^{N+1} + \phi_{j-1}^{N+1} \right)_i^{M+1}$$

$$C_i^M = \left[ \frac{1 - M_R^2 (1 + \mu \sin \psi)^2}{\delta^{2/3}} - (1 + \gamma) M_R^2 (1 + \mu \sin \psi) \left( \frac{\phi_{i+1}^M - \phi_{i-1}^M}{2\Delta x} \right) \right]$$

$$\epsilon_i = 1, \quad \epsilon_{i-1} = 1 \quad \text{for } M_L < 1$$

$$\epsilon_i = 0, \quad \epsilon_{i-1} = 0 \quad \text{for } M_L > 1$$

$$\epsilon_i = 0, \quad \epsilon_{i-1} = 1 \quad \text{for } M_L = 1$$

$$\epsilon_i = 1, \quad \epsilon_{i-1} = 0 \quad \text{for shock point}$$

The epsilon notation denotes that backward and central differences in the x-direction are used in supersonic and subsonic regions, respectively. The use of mixed differences is required for stability. Also, this notation indicates that the sum of the central and backward operators are used at a shock point. This is necessary in order to satisfy the equations in the global sense and insure proper shock jump conditions.

The differencing of  $\phi_{yy}$  is modified on the body as follows:

$$\phi_{yy} = \frac{2}{\Delta y^2} (\phi_{i,2} - \phi_{i,1} - \Delta y \phi_y) \quad (6)$$

For the lifting problem, the potential must be discontinuous in the wake of the rotor. The difference in potential at this discontinuity is equal to the circulation,  $\Gamma$ . In order to difference across this discontinuity,  $\phi_{yy}$  differencing is modified as follows:

$$\phi_{yy}^{\pm} = \frac{\phi_{i,2} - 2\phi_{i,1} + (\phi_{i,-1} \pm \Gamma)}{\Delta y^2}$$

where

$$\Gamma = \frac{C_L}{2\delta^{2/3}}.$$

Differencing, as above, yields a set of nonlinear algebraic equations which must be solved by iteration. In the course of iteration, one can specify either  $C_L$  and iterate to find the angle of attack ( $\alpha$ ), or specify  $\alpha$  and iterate to find  $C_L$ . In the former case,  $\alpha$  must be updated until the jump in potential at the trailing edge equals  $\Gamma$ . This is done using

$$\alpha^{N+1} = \alpha^N - \lambda [\phi^T(x_{te}, 0) - \phi^B(x_{te}, 0) - \Gamma] \quad (7)$$

where  $\lambda$  is a relaxation factor (usually equal to one).

3. SOR METHOD OF SOLUTION. For the SOR scheme, the potential of each iteration is updated using

$$\phi_{i,j}^{N+1} = \phi_{i,j}^N - \omega \frac{R_{i,j}^N}{D_{i,j}} \quad (9)$$

where  $R_{i,j}^N$  is the residual that is defined by eq. (5), with all terms grouped on the right-hand side.  $D_{i,j}$  is the sum of the diagonal elements of the residual:

$$D_{i,j} = \frac{2\epsilon_i C_i}{\Delta x^2} + \frac{(1 - \epsilon_{i-1})}{\Delta x^2} C_i - \frac{2}{\Delta y^2} - \frac{A}{\Delta t^2} - \frac{B}{\Delta x \Delta t} \quad (10)$$

The relaxation factor  $\omega$  is less than 1 in supersonic regions and between 1 and 2 for subsonic regions.

In order to start the problem, the first two time steps are assumed to be quasi-steady (A and B are set equal to zero). This procedure causes no problems as long as the starting Mach number is subcritical.

4. DOUGLAS-GUNN ADI METHOD OF SOLUTION. With this method, it has been found necessary to include an artificial  $\phi^\tau$ -like term in eq. (5) for stability purposes. We now have the modified difference equation

$$\begin{aligned} \frac{F}{\Delta t} \left( \phi^{N+1}_{M+1} - \phi^M \right)_{i,j} + A \delta_{tt} \phi^{N+1}_{M+1} + B^{N+1} \delta_{xt} \phi^{N+1}_{M+1} \\ = \epsilon_i C^M_i \delta_{xx} \phi^{M+1}_{M+1} + (1 - \epsilon_{i-1}) C^M_{i-1} \delta_{xx} \phi^{M+1}_{M+1} + \delta_{yy} \phi^{N+1}_{M+1} \end{aligned} \quad (11)$$

We have not determined the optimal value of  $F$ . However,  $F = 40B$  seems to work well for most uses. For convenience, eq. (11) is rewritten in an operator notation as

$$(KI + \eta \delta_x + S_1 + S_2) \phi^{N+1}_{M+1} + \sum_{\ell=0}^L T_\ell \phi^{N-\ell} = g^{N+1} \quad (12)$$

where

$$g^{N+1} = \frac{F}{\Delta t} \phi^M$$

$I$  = identify operator

$$K = \frac{F}{\Delta t} + \frac{A}{\Delta t^2}$$

$M$  = iteration index

$N$  = time index

$S_1$  = spatial operator in the x-direction =

$$-C^M_i \epsilon_i \delta_{xx} \phi^{M+1}_{M+1} - C^M_{i-1} (1 - \epsilon_{i-1}) \delta_{xx} \phi^{M+1}_{M+1}$$

$S_2$  = spatial operator in the y-direction =  $-\delta_{yy} \phi^{N+1}_{M+1}$

$$T_0 = -\frac{2A}{\Delta t^2} - \frac{B^{N+1}}{\Delta x \Delta t} \delta_x$$

$$T_1 = \frac{A}{\Delta t^2}$$

$$\delta_x = \phi_i - \phi_{i-1}$$

$$\eta = \frac{B^{N+1}}{\Delta x \Delta t}$$

The Douglas-Gunn scheme (ref. 2) is employed to generate the ADI routine because it is generalizable to any number of dimensions. Note that the term  $(KI + \eta \delta_x + S_1 + S_2)$  contains all the unknowns. The original Douglas-Gunn scheme does not include the  $\delta_x$  operator because they did not consider equations that

had mixed spatial-time derivatives. The general Douglas-Gunn recursion relation for generating ADI schemes for any number of dimensions is

$$(KI + \eta \delta_x + S_1) \phi_i^{N+1} + \sum_{j=1}^{i-1} S_j \phi_j^{N+1} + \sum_{j=i+1}^q S_j \phi_j^{N+1} + \sum_{\ell=0}^L T_\ell \phi_i^{N-\ell} = g^{N+1} \quad (13)$$

where  $q$  is the number of dimensions, which is 2 for our case, and  $\phi^*$  is some extrapolation from previous time steps. Generally, we have let  $\phi^*$  equal the previous iterant. The first step of the two-dimensional case is as follows:

$$(KI + \eta \delta_x + S_1) \tilde{\phi} + S_2 \phi^* + \sum_{\ell=0}^1 T_\ell \phi^{N-\ell} = g^{N+1} \quad (14)$$

The coefficient of  $\tilde{\phi}$ , having both central and backward differences, gives rise to either a central or lower tridiagonal matrix. This matrix is solved by a combination of the Thomas algorithm and direct elimination. Substituting in eq. (13) for the second step ( $q = 2$ ), we get

$$(KI + \eta \delta_x + S_2) \phi^{M+1} + S_1 \tilde{\phi} + \sum_{\ell=0}^1 T_\ell \phi^{N-\ell} = g^{N+1} \quad (15)$$

Subtracting eq. (14) from eq. (15), yields a simplified step 2, which is as follows:

$$(KI + \eta \delta_x + S_2) \phi^{M+1} - (KI + \eta \delta_x) \tilde{\phi} - S_2 \phi^* = 0 \quad (16)$$

KI and  $S_2$  generate a tridiagonal matrix. However,  $\eta \delta_x$  adds a term which is on a previous column. This column must be solved before we can move to the next column. This imposes a necessary order on the process and the tridiagonal inversions march across the grid from left to right. To get started, we let the first two time steps be quasi-steady by setting the time index ( $n$ ) equal to the iteration index ( $m$ ).

**5. RESULTS.** Numerous cases were run by both the SOR and ADI methods, using various Mach number and lift variations. The greatest difference between these calculations and the results of ref. 1 is the inclusion of lift. It was noted in ref. 1 that upstream wave propagation necessitated placing the upstream boundary at least 10 chords upstream. With lift, however, this situation becomes much more acute, and it was necessary to place the boundaries 20 chords away (upstream and downstream) in order to obtain stable solutions. The ADI method was generally more reliable, as the SOR method often failed to converge at higher values of lift and Mach number. However, when the SOR method did converge, it agreed almost exactly with the ADI method. In these situations, however, the ADI method appeared to be about twice as fast as the SOR.

A case, from which identical results were obtained from both methods, is shown in fig. 1. In this case, we specify a sinusoidal  $C_L$  and find the angle of attack. The rotor in the center of fig. 1 indicates the various azimuth angles for the  $C_p$  plots.

Unsteadiness is indicated by the effect of flow history on the flow. For example, at the rotor positions A and I, the Mach number, angle of attack and lift are the same. Also, the solutions obtained are so identical that they are plotted on top of each other. Clearly, the flow history plays no role at these points and the flows are essentially steady. At blade position D, the lift is specified to be zero, and the solution obtained has identical flows on the top and bottom surfaces. This is what one expects from steady flows. However, one would also expect a zero angle of attack from a steady solution instead of the slightly negative angle that was actually obtained. The negative angle implies that the flow, up to this point, is unsteady but only slightly so.

In the subsequent blade locations, the flow is decelerating and the lift is again increasing. The shock is seen to move forward but does not diminish much in strength. In fact, on the bottom surface, the shock does not go to zero strength at all. Instead, it appears at position G to be on the verge of popping off the leading edge and proceeding out into space. This effect was first noted in ref. 1 for nonlifting cases and is seen with great clarity in ref. 3, which uses an ADI technique very similar to this one. On the top surface, the increasing lift allows for the sustenance of the supersonic region, and there is no shock popping. However, the shocks are substantially stronger than at their corresponding azimuths in the accelerating flow region. For example, position H is the mirror image of position B, but the  $C_p$  distributions for the upper surface are quite different.

Clearly, the effect of unsteadiness is large in a decelerating transonic flow. In addition, where there is lift, the difference in unsteadiness can be quite different on the top and bottom surfaces.

6. CONCLUDING REMARKS. The modified Douglas-Gunn ADI is about twice as fast as SOR and generally more reliable. Present computing times are about 20 minutes on a CDC 7600 for a two-dimensional lifting problem. This can be greatly improved with more efficient programming, a less rigid convergence criterion, and the use of a variable time step. An interesting approach is taken in ref. 3, where the nonlifting two-dimensional problem is treated, using a Douglas-Gunn scheme. In this approach, the difference equation is linearized in time, thus eliminating the iterations, but requiring a smaller time step. It appears that with a combination of the above approaches, unsteady three-dimensional calculations would be a practical proposition. We are now working toward this end.

It appears that the inclusion of unsteadiness in transonic flow prediction is at least as important (and likely more important) than the effect of three dimensionality. Decelerating flows are altogether different from their quasi-steady counterparts. Also, lift raises the possibility of very unusual loads in the second rotor quadrant.

#### *Acknowledgment*

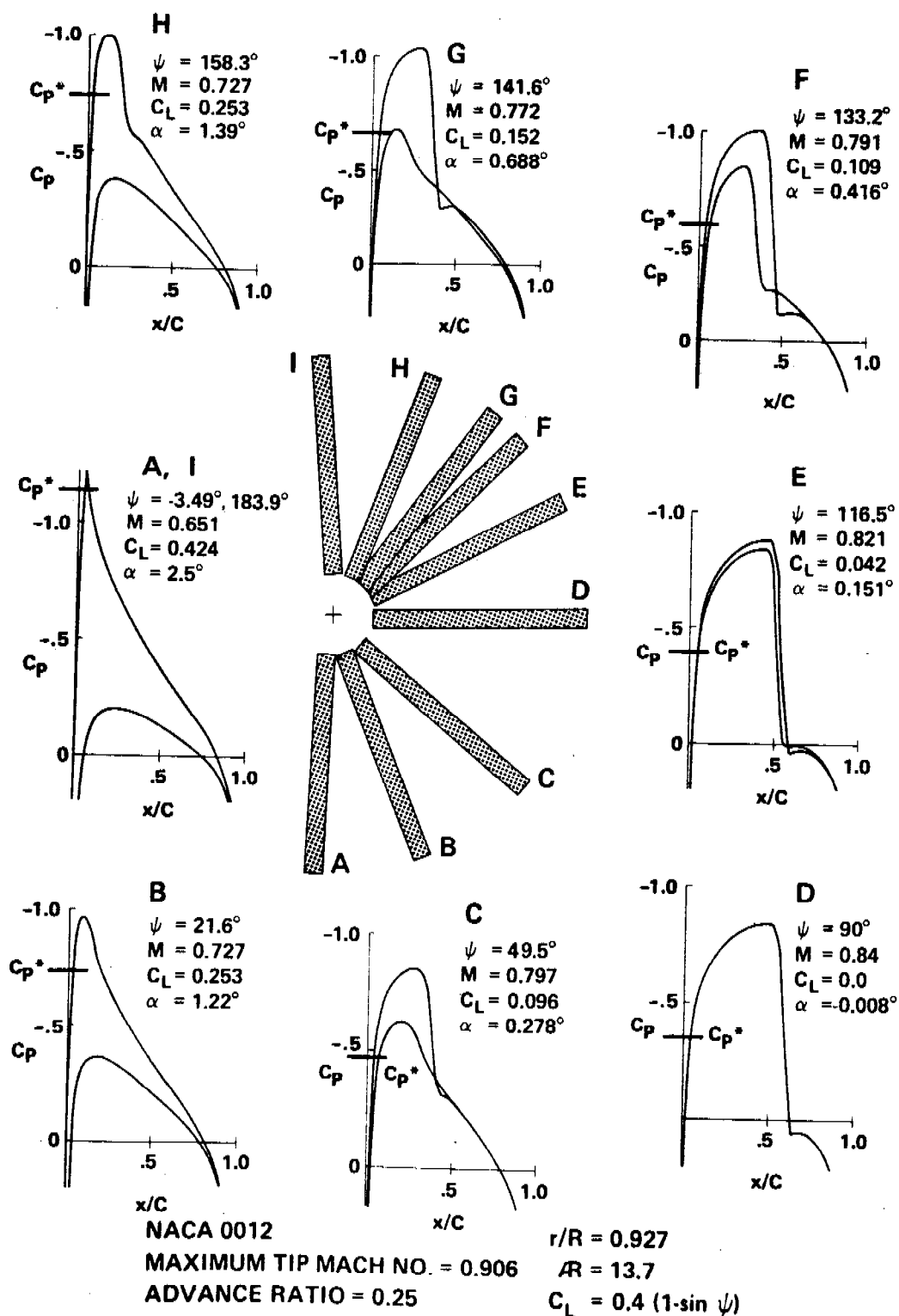
The authors gratefully acknowledge the valuable contributions of Francis X. Caradonna to this work.

## REFERENCES

1. Caradonna, F. X., and Isom, M. P.: Numerical Calculations of Unsteady Transonic Potential Flow Over Helicopter Rotor Blades, AIAA Paper 75-168, presented at AIAA 13th Aerospace Sciences Meeting, Pasadena, Calif., Jan. 20-22, 1975.
2. Douglas, J., and Gunn, J. E.: A General Formulation of Alternating Direction Methods, Numer. Math., Vol. 6, 1964, pp. 428-453.
3. Ballhaus, W. F., and Steger, J. L.: NASA TM X-73,082, Nov. 1975.



# CALCULATED TWO-DIMENSIONAL PRESSURE DISTRIBUTIONS ON A ROTOR





## UTILIZING REAL-TIME TEST DATA ANALYSIS IN SYSTEM MONITORING AND CHECKOUT

E. H. Gamble  
U. S. Army Test and Evaluation Command  
Deputy to the CG for Analysis  
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. For the proper utilization of equipment needed for test data on-line acquisition, display, and comparative analysis, some novel data management techniques are in the process of development.

For on-line analysis of test display data, the recognition of changing values in the inherent uncontrolled design parameters and system output variables must be basic. Also, the display technique must accommodate a reasonable number of sets of operating and environmental conditions.

We must be able to identify shifts in the values of the parameters and variables as well as shifts in the weighted contributions of each parameter to the output system variable value. Each contribution is the product of the quantified parameter value under a given condition set and the corresponding influence coefficient. This coefficient reflects the sensitivity of the output variable to a nominal change in the parameter when measured at the given condition set. In addition, we must identify any changes in the gains of the functional equipment components and variations in the transmission characteristics of the test data transport and management system. An observation matrix of the coefficients, parameter values, or contributions provides the display vehicle.

The study adopts a model consisting of seven parameters and five sets of conditions. Variations of all quantities are demonstrated and companion computations made to show the significance of the variations and their identification.

## I. INTRODUCTION.

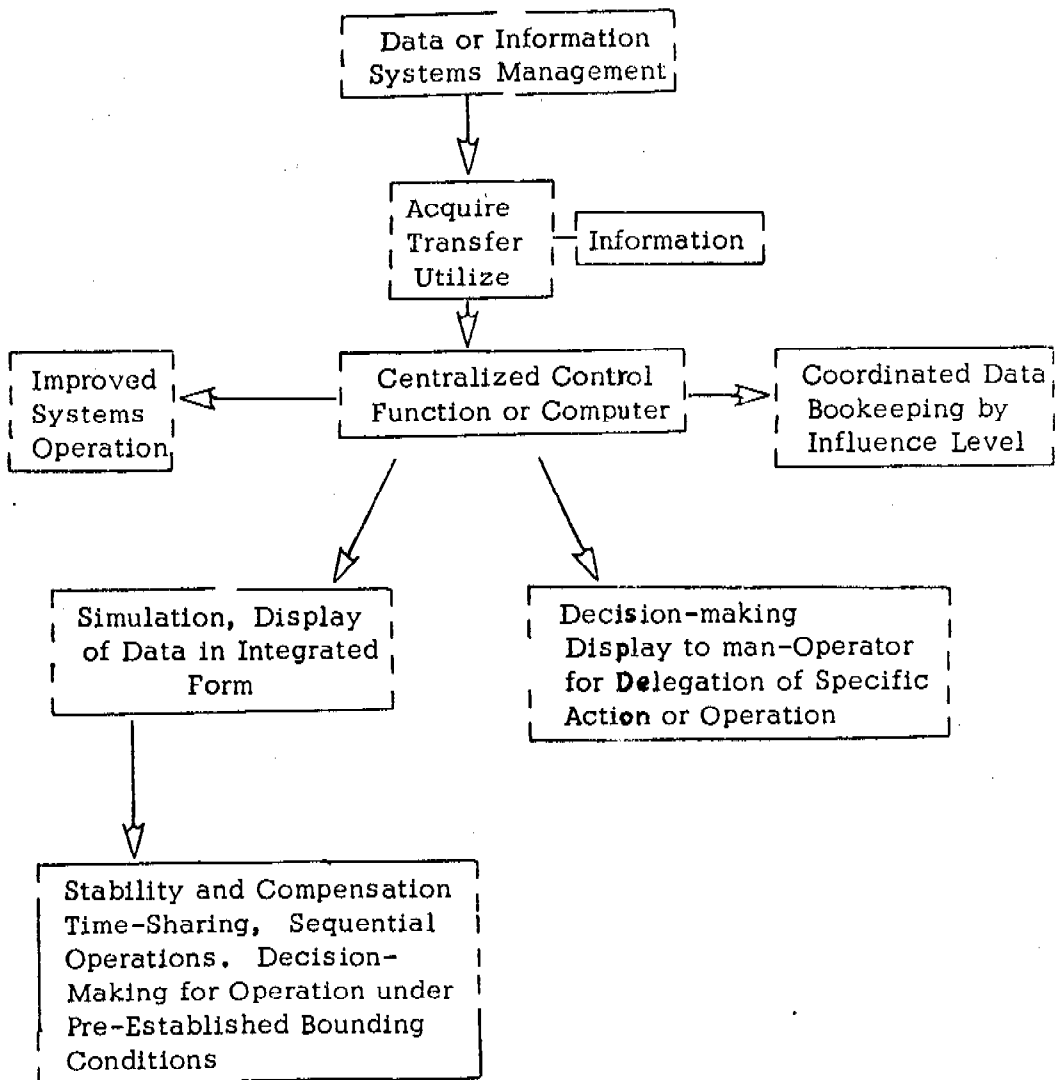
One of the significant developments in process to be applied to the test analysis and evaluation of Army materiel systems is the real time use of the computer for the:

1. On-line acquisition and analysis of data.
2. Display of the test data while the system is under test.
3. Making comparisons of this test data with stored data from a developed reservoir of historical data about similar and current systems. Knowledge of the testing subsystem equipment should be available for ready reference (see Figure 1).

One such testing system is in the design and planning stage and called project ADAPT (Automatic Data Acquisition and Processing Technology). This program is under the direction of the Materiel Testing Directorate of the Aberdeen Proving Ground.

A practical technique for on-the-spot analysis of test display data is needed. For such on-line analysis, not only central or "expected" values for the parameters must be displayed at the centralized control and display area. We must identify shifts in the values of the uncontrolled parameters of the system under test and the testing and instrumentation system. In addition, we must identify any changes in the gains of the functional components of equipment and the variations in the transmission characteristics of the data transport and management system.

To assist in the understanding of utility and potential application for such a system, a study is described where the data display is evaluated as a means for both real time system monitoring and confidence status or checkout. A display model for the test item and the testing equipment will be applied to a planned group of activities. These activities are defined as the set of subtests planned and executed as a part of the Test Plan and defined by a Test Directive. One approach to this model is the adoption of a Data Observation Matrix as the display vehicle. This model has found use wherever one applied techniques such as systems modeling by functions (see Reference 1) on parameter-sensitivity coefficients (see Reference 2). Figure 2 describes a general form for such an observation matrix.



Pay-Off Portrait of Large Systems Data Management

Figure 1

$i$ or $j$	$P_j$ or $P_j$	Measured Parameter Values	K mission phase for k subsystem
$(X_i)_k$ or $(X_j)_k$ Stimuli or Subsystem Condition	$a_{ijk}  _{k \text{ t l m n o g fixed}}$ or $a_{ijk}  _{k \text{ t l m n o g}}$		$(w_{ki})_k = \sum_j a_{ijk}(P_j)_k$  Display Values
	$(\beta_j)_{kk} = \sum_{i=1}^{m_j} (a_{ijk})_k$		$\sum_i (w_{ki})_k = \sum_j \beta_j(P_j)^0$

### INFLUENCE MATRIX

Figure 2

It will be our purpose to illustrate the application of analysis techniques for a matrix model of seven parameters and five condition sets of grouped activities. The complete study of this model involves the display for the time variations in the participating parameters and their dimensionalized contributions and changes in the parametric sensitivities.

## II. DISCUSSION.

### A. Mathematical Background.

#### Utility of Observation Matrix for Data Display:

One evaluation criterion for usefulness for any method of information identification and cataloging is that the variations of all system parameters be recognizable, definable, and predictable. We have suggested that the mathematical and informational system model for a given influence level be described by the coefficients array for the applicable engineering equipment and physical constraints.

The influence level is defined by the operating subsystem or data system gross functions,  $K$  or mission phase, the data functional model subfunction or component, activity and agency, etc. The  $(p_j)$  are the measured values for the  $X_j$  parameters or the set of values found as the set to maximize or minimize the criterion - our measure of adequacy/excellence.

Two cases are of immediate interest. For a change in system time of magnitude  $(\Delta t)$ :

(1) A shift occurs in the system parameter measurable values (polarity included).

(2) The shift occurs in the influence coefficients, which represent either the compatibility potentials or sensitivity potentials of the equipment. Transfer or data transmittal characteristics are the inter-dependency factors for the variables and the system conditions.

In both cases, the display integrated parameters and contributions to the system objective functions are changed in magnitude. A dependable technique must be established for determining whether the shift is made up

of type (1), (2), or some combination of these. Variation with system condition/configuration (J) should be identifiable.

A numerical example with seven parameters and five conditions or configurations which define the interdependence relationships will be used to illustrate both cases (1) and (2). The example will also show a variation with (J) for one of the seven parameters. We will study these for a system time,  $t = t_1$ , and a time change  $\Delta t_1$ , a second time  $t_2$  and time change  $\Delta t_2$ . The data are available for a consideration of the variations consistent with the time interval  $(t_2 - t_1)$ , as well as those mentioned.

For Case (1), where the shift occurs in the parameter values, the supporting analysis follows:

The change in objective criterion equals --

$$\Sigma^* - \Sigma = \beta_1 (x_1^* - x_1) + \dots + \beta_n (x_n^* - x_n) \quad (.01)$$

The corresponding change in subsystem capacity potential or integrated parameter is

$$w_{Ki}^* - w_{Ki} = a_{1Ki} (x_1^* - x_1) + \dots + a_{nKi} (x_n^* - x_n) \quad (.02)$$

$$\Sigma^* - \Sigma = \sum_i \left\{ w_{Ki}^* - w_{Ki} \right\} \quad (.03)$$

$$= \sum_i^m \sum_J^n a_{iKJ} (x_J^* - x_J) = \sum_J \beta_J (x_J^* - x_J) \quad (.04)$$

$$= \sum_{i=1}^m \sum_{J=1}^n (a_{iKJ}) (\delta p_J) = \sum_J (\beta_J) (\delta p_J) \quad (.05)$$



(Note: The starred quantities (previous page) are values for variables at  $(t_1 + \Delta t)$  and unstarred those at  $t_1$ .)

A check test is performed in which all the stimuli parameters  $X_J$  are set at a fixed constant value (normalized, this is equivalent to unity, 1). If this will give a zero change in the objective function  $\Sigma = \Sigma^*$ , the valid conclusion is that there have been no changes in the influence coefficients  $\{a_{iKJ}\}$ . Superscript (1) identifies check values.

If both  $(\sum_J^* - \sum_J)$  and  $\sum_J (\beta_{JK}^* - \beta_{JK})$  have a value which is

non-zero for the test of fixed constant values for the parameters, then the variations are in the  $(a_{iKJ})$ . Mathematically, for the change in the coefficients, we state that:

$$\sum_J^{(1)*} - \sum_J^{(1)} = \sum_J^{(1)} \sum_i^{(1)} (a_{iJK}^* - a_{iJK}) \geq 0 \quad (.06)$$

$$= \sum_J^{(1)} (\beta_J^* - \beta_J) \geq 0; \quad X_J = P_J^* = P_J = 1 \quad (.07)$$

For a shift in the parameter values, we state that:

$$\left[ X_J^* - X_J \right] = \left( \Delta X_J \right) \Big|_{\Delta t} = \left( \frac{\left( \frac{\beta_J^* X_J^*}{\beta_J \cdot 1} \right)}{\left( \frac{\beta_J X_J}{\beta_J \cdot 1} \right)} \right) \Big|_{\Delta t} \quad (.08)$$

for  $X_J^* \neq X_J = P_J = 1$

The time sensitivity potential for the parameter  $X_J$  is found by dividing the expression and value for  $\Delta X_J$  for the time shift  $\Delta t$

by the time interval  $\Delta t$ . This sensitivity is identical to the time rate of change of the variable  $X_J$ .

$$\left( \frac{\Delta X_J}{\Delta t} \right)_K^{\Delta t \text{ change}} = \left. \frac{\partial X_J}{\partial t} \right|_{t,K} \quad (.09)$$

$$= \frac{(\Delta t)^{-1}}{(\beta_J^* \beta_J)_{(X_J=1)}} \left\{ \beta_J^* \beta_J^{(1)} X_J^* - \beta_J^{(1)} \beta_J X_J \right\}_K \quad (.10)$$

$$= \frac{(\Delta t)^{-1}}{(\beta_J^*)} [\beta_J^* (1 + \delta P_J) - \beta_J] \quad (.11)$$

When the change is in the influence coefficients and not in the  $X_J$  parameters, we must answer the selection question regarding equipment deterioration. "Do we need to determine the individual change  $\Delta(a_{iJK})$  or the resultant effect?" Let us consider the mathematical forms needed to answer this question.

$$\Delta \left( \sum_i a_{iJK} X_J \right)_{K,K} = \Delta \left( \beta_J X_J \right)_K = X_J (\Delta \beta_J) \quad (.12)$$

$$\Delta w_{Ki} = \Delta \left( \sum_J a_{iJK} X_J \right) = \sum_J (\Delta a_{iJK}) X_J \quad (.13)$$

for  $\Delta X_J = 0$

$$\Delta \sum_K = \left( \sum^* - \sum \right)_K \quad (.14)$$

For the particular case where  $X_J = 1$ ,

$$\left( \sum^* - \sum \right)_K = 0 \quad \text{for } \Delta a_{iJK} = 0 \quad (.15)$$

$$\leq 0 \quad \text{if } \Delta a_{iJK} \neq 0 \quad (.16)$$

The individual ( $\Delta a_{iJK}$ ) are needed in order to determine or verify the change in the subsystem output variable to be derivable from the summed weighted influence of the parameters (see Equation 13). Fortunately, our matrix technique displays all these quantities and they are recognizable.

For our Case (2), where the shift is in the coefficients and not the parameters, we know these facts:

$$\sum_i^* - \sum_i = \sum_i \left( w_{Ki}^* - w_{Ki} \right) \quad (.17)$$

$$= \sum_J X_J \sum_i \left( a_{iJK}^* - a_{iJK} \right) \quad (.18)$$

$$= \sum_J X_J \sum_i (\Delta a_{iJK})$$

$$= \sum_J X_J \left( \beta_J^* - \beta_J \right)_K \quad (.19)$$

$$= \sum_J X_J (\Delta \beta_J)_K .$$

Since our variations with  $\Delta t$  are in  $a_{iJK}$ , we set  $X_J$  equal to plus one, then we find:

$$\sum_i^{(1)*} - \sum_i^{(1)} = \text{value}_K = \sum_J \left( \beta_J^{(1)*} - \beta_J^{(1)} \right)_K = \sum_J (\Delta \beta_J) \geq 0 \quad (.20)$$

$$\begin{aligned} &= \sum_J \sum_i \left( a_{iJK}^* - a_{iJK} \right)_K \\ &= \sum_J \sum_i (\Delta a_{iJK}) . \end{aligned} \quad (.21)$$

A priority ranking can be found from this ratio:

$$\begin{aligned}
 & \frac{\left( \beta_J^* - \beta_J \right)_K^{(1)}}{\left( \beta_{(J+1)}^* - \beta_{(J+1)} \right)_K^{(1)}} \\
 &= \frac{(\Delta \beta_J) X_J = 1}{(\Delta \beta_{(J+1)}) X_J = 1} \quad (.22)
 \end{aligned}$$

#### B. Illustration of Technique.

We will study a system whose model consists of seven parameters subjected to five constraints. These constraints describe the system parameter interrelationships under five sets of conditions.

Our five system conditions or constraint inequalities form the rows for our Influence Matrix. Seven dominant parameters, which are interdependent, form the headings for the columns in the array. Nine tables are attached which illustrate the application of our technique for identification of the changes in status for the operating equipment. Let us consider one system situation which could be represented by the coefficient array of Figure 2. The integrated subsystems may be described by the seven dominant and critical parameters whose measured values are available and accessible as the outputs of the several position references, inputs to the circuits for the power controller, and inputs to the path control circuits. The five system conditions might define the several constraints for a land vehicle during the conduct of a particular set of maneuvers.

Table 1 has cell values of  $a_{ijk}$  at the system operating time  $t_1$ . These coefficients are the interdependence transfer admittances or compatibility potentials. The  $C_{Ki}$  are the integrated display parameters for the several system conditions and represent the limitation on the seven term sum of the product of the interdependence coefficients and the appropriate stimuli or dominant parameters (for the specialized case of unity value for each of these dominant parameters). The  $b_j$  are the corresponding

SENSITIVITY - INFLUENCE TABLE FOR SHIFT IN PARAMETER VALUES WITH ( $\Delta t$ )

$J \rightarrow$ 1 $\downarrow$	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$C_{Ki} (P_J = 1)$
$A_1$	863	450	130	445	180	550	190	2808
$A_2$	804	450	130	445	180	95	190	2124
$A_3$	960	675	155	525	180	120	200	2623
$A_4$	450	450	104	195	72	50	100	1385
$A_5$	720	450	86	455	151	100	180	1925
$b_J$ at $t_1$	3797	2475	605	2075	771	915	940	$\left( \sum_J b_J   (P_J = 1) \right)_K = 11574 = \sum_I C_{Ki}$
$P_J (t_2)$	1.903	0.641	1.000	0.738	0.406	0.861	0.515	
$P_J (t_2 + \Delta t_2)$	2.135	0.898	0.593	0.896	0.056	0.166	0.095	
$\Delta P_J (\Delta t_2)$	0.230	0.257	0.407	0.158	0.350	-0.695	-0.420	
$\Delta P_J [\Delta t_2 + (t_2 - t_1)]$	0.128	0.311	0.385	0.014	-0.365	-0.682	+ 0.428	
$P_J (t_1)$	2.007	0.587	0.208	0.882	0.421	0.848	0.523	
$b_J P_J$ at $t_1$	7592	1452	126	1829	322	776	491	$\left( \sum_J b_J P_J \right)_K = 12588$ at $t_1$

TABLE 1

capacity potentials for the unity measured parameter values. This table gives the parameter values for three system times--  $(t_1)$ ,  $(t_2)$ , and  $(t_2 + \Delta t_2)$ .

Table 2 gives the array of cell values at system time  $(t_2)$  and each is the contribution at the corresponding  $i$  system condition to the display parameter. In addition, an optimization study was performed for the given array to find the set of weighting coefficients or optimal set  $(p_j^*)$  for a maximization-minimization of the criterion. This set has significance for an overall performance evaluation of the operating subsystem rather than a status or diagnosis study.

Table 3 provides the array of display contributions at time  $t_1$ . These values present rather obvious changes in cell values from those shown in Table 2.

Table 4 portrays the contributions at system time  $(t_2 + \Delta t_2)$  in contrast with those identified as a part of Table 2 at  $(t_2)$ .

An optimization study was also performed at this system time to find the set  $(p_j)^*$ .

Table 5 gives the variations in the display contributions for the time change of  $(\Delta t_1)$ . This array is used to illustrate our Case (1) where the shift occurs in the parameter measured values with no changes in  $(a_{ijk})$ . The change in the integrated display parameter  $w_{ki}$  is given for each  $i$  system condition. The percentage change in the parameter value measured at  $t_1$  is also determined. This set of values shows the significant dominance of  $X_j$  for  $j = 1, 2$ , and 4.

Table 6 gives the variations in the display contributions for the time change of  $(\Delta t_2)$ . The  $p_j$  corresponding to the  $X_j$  are considered fixed. The change in  $\sum_i \Delta w_{ki}$  is a positive quantity 282. However, the summation  $\sum_j (\Delta \beta_j)$  is the negative quantity (-561).

Table 7 is a companion array to Table 6, but giving the changes in  $a_{ijk}$  rather than  $\delta[a_{ijk}(p_j)]$ , the variation, or  $(p_j) \delta[a_{ijk}]$ . The summation of the variations in  $a_{ijk}$  first with respect to  $i$ , and then  $j$  is a negative quantity (-540). This array is numerically identical with the test

TABLE OF CONTRIBUTIONS  $(a_{iJK} P_J)$  AT  $t = t_2$

$i \downarrow J \rightarrow$	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$w_{Ki}$	$f_{Ki}$
$A_1$	1644	289	130	328	73	473	98	3035	3322
$A_2$	1531	289	130	336	73	82	98	2539	2832
$A_3$	1826	433	155	387	77	103	103	3084	3418
$A_4$	857	287	104	144	31	43	52	1518	1666
$A_5$	1371	289	86	336	61	86	93	2322	2553
$\beta_J$	7231	1589	605	1531	316	787	444	$\sum \beta_J = 12503, 12500 = \sum w_{Ki}$	
$(P_J)^*$	1.346	0.650	1.312	0.738	0.612	0.880	0.490	$\sum_i f_{Ki} = 12790 = \sum_J \left( \beta_J P_J \right)^*$	
$(\beta_J P_J)^*$	9732	1032	794	1129	193	692	218		

Individual contribution to  $w_{Ki}$  is  $\left( a_{iJK} \right) P_J (t_2)$   
 to  $f_{Ki}$  is  $\left( a_{iJK} P_J(t_2) \right)^{P_J^*}$

$(P_J)^*$  are optimum values of parameters for the matrix array having individual cell designations  $a_{iJK} P_J (t_2)$ . The result from an iterative, convergent process maximizing-minimizing the criterion  $\sum_i f_{Ki} - \sum_J \left( \beta_J P_J \right)^*$

TABLE 2

VALUES OF  $(a_{iJK} p_J)^{**}$  AT  $(t = t_1)$

$i \downarrow$	$J \rightarrow$	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$(w_{Ki})_k$
	$A_1$	1726	264	27	392	76	466	99	3050
	$A_2$	1606	264	27	401	76	81	99	2554
	$A_3$	1920	396	32	463	76	102	105	3094
	$A_4$	900	264	22	172	30	42	94	1524
	$A_5$	1440	264	18	401	64	85	94	2366
	K III	7592	1452	126	1829	322	776	491	$\sum (w_{Ki})_k = 12588$

$$\sum_{i=1}^5 (a_{iJK} p_J)^{**} \text{ at } t = t_1 = \text{K III}$$

TABLE 3



TABLE OF CONTRIBUTIONS AT  $t = t_2 + \Delta t_2$

Values at  $t = t_2 + \Delta t_2$

i ↓	J →	1	2	3	4	5	6	7	$W_{ki}$	$F_{ki}$
	$A_1$	1842	405	77	398	10	91	17	2840	2911
	$A_2$	1716	405	77	408	10	16	17	2649	2720
	$A_3$	2048	607	92	470	10	20	18	3265	3349
	$A_4$	960	405	62	175	4	8	9	1620	1672
	$A_5$	1546	405	51	408	8	17	16	2441	2481
	$\beta_J^*$	8102	2227	359	1859	42	152	84	$\sum_J (\beta_J^*)_k = 12815 = \sum_i W_{ki}$	
177	$\beta_J^* P_J^{***}$	9641	2062	217	903	4	93	121	$\sum_i F_{ki} = 13027 = \sum \beta_J^* P_J^{***}$	
	$P_J^{***}$	1.1899	0.926	0.605	0.486	0.0846	0.613	1.437		

Individual contribution to  $W_{ki}$  is  $\{a_{1JK}\}_k P_J^{***}(t_2 + \Delta t_2)$

to  $F_{ki}$  is  $\{a_{1JK} P_J(t_2 + \Delta t_2)\} P_J^{***}$

$\{P_J^{***}\}$  = the parameter value set resulting from optimizing  $\sum_i F_{ki} = \sum_J \left( \beta_J^* P_J^{***} \right)_k$

TABLE 4

# VARIATION IN DISPLAY PARAMETERS

Element is  $\delta(a_{iJK} P_J) \approx a_{iJK} (\Delta P_J)$  for  $\Delta t_1$

$\downarrow i$	$J \rightarrow$	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$(\Delta w_{ki})$
$A_1$		116	141	50	6	-66	-375	-82	-210
$A_2$		110	141	50	7	-66	-65	-82	+95
$A_3$		128	211	60	7	-66	-82	-87	+171
$A_4$		60	141	40	3	-26	-34	-78	+106
$A_5$		96	141	33	7	-56	-68	-78	-75
$(\Delta K) = (K_J^* - K_J)$		510	775	233	30	-280	-624	-407	$\sum = 237$
$b_J$		3797	2475	605	2075	767	915	940	$\sum (b_J) = 11574$
$\frac{1}{88} (\Delta P_J) = (P_J^* - P_J)$		0.128	0.311	0.385	0.014	-0.682	0.428	-.033	$J$

$$\sum_i (w_{Ki}^* - w_{Ki}) = \sum_J b_J (P_J^* - P_J) = \sum_J (K_J^* - K_J)_k$$

No change assumed in  $a_{iJK}$  from  $t_i$  to  $(t_1 + \Delta t_1)$

Data obtained as differences in Tables 3 and 4

$\left( \frac{\Delta P_J}{P_J} \right)$	=	.064	0.53	1.85	0.016	-0.87	-0.804	+0.82
		6.4%	53%	185%	1.6%	-87%	-80.4%	+82%

TABLE 5

VARIATIONS IN EQUIPMENT INFLUENCE COEFFICIENTS. Element is  $\delta(a_{iJK} \quad P_J)$

Change for  $(\Delta t_2)$  from Tables 2 and 4.

$$= (t_2 + \Delta t_2) - t_2$$

179

$i \downarrow$	$J \rightarrow$	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$\Delta^{w_{Ki}}$	$\frac{\Delta^{w_{Ki}}}{P_J(t_2)}$	
	$A_1$	198	116	-53	70	-63	-382	-81	-195	-428	
	$A_2$	185	116	-53	72	-63	-66	-81	+110	-66	
	$A_3$	220	174	-63	83	-67	-83	-85	+179	-61	
	$A_4$	103	116	-42	31	-27	-35	-77	+69	+47	
	$A_5$	165	116	-35	72	-53	-69	-77	+119	-31	
	$K_J^* - K_J = \Delta^K(\Delta t_2)$	871	638	-246	328	-273	-635	-401	$\sum_J \Delta^K = 282 = \sum_i (\Delta^{w_{Ki}})$		
	$P_J(t_2 \text{ to } t_2 + \Delta t_2)$	1.905	0.641	1.000	0.738	0.406	0.861	0.515	$\sum \frac{\Delta^{w_{Ki}}}{P_J(t_2)} = -555$		
	Assume that total shift during $(\Delta t_2)$ is due to changes in $a_{iJK}$										
	$\left(\frac{\Delta^K}{P_J}\right)_J$	457	995	-246	430	-672	-737	-778	$\sum_J \frac{\Delta^K}{P_J} = \sum_J \Delta \beta_J = -561$		

TABLE 6

TABLE OF EQUIPMENT VARIATIONS

		$P_J$ fixed; changes in $a_{iJK}$							Period = $(\Delta t_2)$		
$i$	$J \rightarrow$	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$\sum_J \delta a_{iJK}$	$\Delta w_{K1}$	$\frac{\Delta w_{K1}}{\sum (\delta a_{iJK})}$
180	$A_1$	104	181	-53	95	-155	-443	-157	-428	-195	0.45
	$A_2$	97	181	-53	98	-155	-77	-157	-66	+110	-1.66
	$A_3$	115	271	-63	42	-164	-97	-165	-61	+179	-2.93
	$A_4$	54	181	-42	111	-66	-41	-150	147	+69	+1.47
	$A_5$	86	181	-35	98	-131	-80	-150	-31	+119	-3.84
	$(K_J^* - K_J) = (\Delta K)_J$	871	638	-246	328	-273	-635	-401			
	$(\sum_i \delta a_{iJK}) = \delta a_{KJ}$	456	995	-246	444	-672	-738	-779	$\sum_J (\sum_i \delta a_{iJK})$	= -540	
	$P_J(t_2)$	1.905	0.641	1.000	0.738	0.406	0.861	0.515			
Values of Table 6 contributions changes divided by $p_J$ to form $(\delta a_{iJK})$											
Priority Ranking of Deterioration		5	1	7	6	4	3	2	Equals $(B_J^* - B_J) / (B_{J+1}^* - B_{J+1})$		

TABLE 7

condition of setting all of the parameter measured values to the same constant -- unity. The criterion value (-540) is then the  $\sum_j (B_j^* - B_j)$ .

A priority ranking can be found from the ratio of each  $(B_j^* - B_j)$  to the next parameter  $B_{j+1}^* - B_{j+1}$ .

Table 8 gives the data of Table 7 now with the cell value as the percentage change which has occurred in  $\{a_{iJK}\}$  for the time change  $\Delta t_2$ . These are the shifts or deterioration in the operating characteristics of the equipment defined for their influence corresponding to  $X_j$ .

Table 9 records the array of  $\{a_{iJK}\}$  coefficients for their values after the  $\Delta t_2$  shift has occurred. These are the total deteriorated values for each interdependence cell. Of some significance are the % change in the display parameters (for  $p_j = 1$ ). These cover a percentage change from -18% to + 3.3%. The corresponding changes in the true display parameters with  $P_j(t_2)$  actual values show a range of -6.8% to +5.5%. The change in criterion value  $\sum_j b_j$  for  $P_j = 1$ ,  $t = t_2$  is from 11,574 to 10,326. The data indicates a constant percentage change in equipment transfer characteristics with varying system conditions, except for the one case of ( $J = 4$ ). This latter variation would provide a basis for forecasting a nonlinear type of deterioration with system operating conditions. Such a variation does not lend itself easily to malfunction isolation, identification, and corrective action.

### III. CONCLUSIONS.

The study is a first cut at enlarging our knowledge concerning the applicability and limitations of the Observation Matrix Technique for Data System Cataloging. What we have learned from this study:

1. The matrix technique will permit the identification of variations in display integrated parameters (system quantities  $w_{Ki}$ ) and the individual contributions due to each selected dominant equipment or software parameter. These parameters may be functional in origin -- hence an activity or action.

2. It will permit the separation of the variations into changes in values of the influence or interdependence (interface) coefficients or potentials or changes in the measured values of the parameters. The parameters can also be equipment signals inputs or stimuli.

$j \rightarrow$	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$
Average	12%	40.2%	-40.8%	21.3%	-86%	-85%	-82%
A <sub>1</sub>	12%	40.2%	-40.8%	21.3%	-86%	-85%	-82%
A <sub>2</sub>	12%	40%	-40.8%	22%	-86%	-81%	-82%
A <sub>3</sub>	12%	40.2%	-41%	8.0%	-91%	-81%	-82%
A <sub>4</sub>	12%	40.2%	-40.2%	59%	-91.5%	-81%	-82%
A <sub>5</sub>	12%	40.2%	-40.8%	21.6%	-87%	-80%	-82%
Average	12%	40%	-40%	+26.8%	-88%	-81%	-82%

$\left( \frac{\partial a_{JK}}{\partial p_j} \right)_{j \text{ fixed or } j = K}$

Only  $j = 4$ , shows a marked variation in the  $a_{JK}$  change with  $i$

There is a constant change in equipment transfer characteristics with the shift in equipment stimuli/configuration

conditions, except for ( $j = 4$ ).

$$\frac{\partial a_{JK}}{\partial p_j} \bigg|_{j=j} = 0 \quad \text{for } j = 1, 2, 3, 5, 6, 7$$

Mathematically,

$$\frac{\partial a_{JK}}{\partial p_j} \bigg|_{j=j} \neq 0$$

for  $j = 4$ ,  $p_j$  for  $j = 1, 2, 4$  are dominant variables.

TABLE 8

INFLUENCE MATRIX FOR

$\underline{a}_{iJK}$  @  $t_2$ ,  $(\Delta t_2)$  shift

i ↓	J →	X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	X <sub>4</sub>	X <sub>5</sub>	X <sub>6</sub>	X <sub>7</sub>	$\frac{C_{Ki}}{P_J = 1}$	$\frac{\Delta C_{Ki}}{C_{Ki}}$	$\frac{\Delta w_{Ki}}{w_{Ki}}$
	A <sub>1</sub>	967	631	77	540	25	107	33	2380	-18%	-6.8%
	A <sub>2</sub>	901	631	77	543	25	18	33	2058	-3.2%	+4.1%
	A <sub>3</sub>	1075	946	92	567	16	23	35	2562	-2.4%	+5.5%
	A <sub>4</sub>	504	631	62	306	6	9	30	1432	+3.3%	+4.2%
	A <sub>5</sub>	806	631	51	553	20	20	30	1894	-1.6%	+4.9%

183

$$\begin{aligned}
 & \sum_J b_J(t_2) = 10,326 \\
 (1) \sum_J \delta \underline{a}_{iJK} & \quad \begin{matrix} 456 & 995 & -246 & 444 & -672 & -738 & -779 \\ 3797 & 2475 & 605 & 2075 & 771 & 915 & 940 \end{matrix} \quad \begin{matrix} = -540 \\ = \sum_J \sum_i \delta \underline{a}_{iJK} \end{matrix} \\
 & b_J \quad \begin{matrix} 3797 & 2475 & 605 & 2075 & 771 & 915 & 940 \end{matrix} \\
 (1)/b_J & \quad \begin{matrix} 12\% & 40.2\% & -40.6\% & +21.4\% & -87\% & -81\% & -83\% \end{matrix} \quad \begin{matrix} \frac{-540}{10,326} = -5.2\% \\ = \frac{\sum_J \sum_i \delta \underline{a}_{iJK}}{\sum_J b_J} \end{matrix}
 \end{aligned}$$

TABLE 9

3. The test in which the parameters are replaced by a constant value (normalized to unity) does prove of value in identifying the source of deterioration (equipment operating characteristics  $a_{ijk}$  -- or parameter value).

4. It permits an analysis of the variation as percentage change in  $a_{ijk}$  or parameter values  $P_j$  for  $X_j$ .

5. A variation in the equipment characteristic with system condition -- such as gain, impedance, etc., is identified with stimulus, parameter, and condition.

6. The matrix technique is directly applicable to an optimization analysis which gives a set of parameter stimuli values for comparison with actual measured values. This optimized set  $\{P_j\}$  is the desired parameter set for the several system conditions or configurations. This optimized set may be the desired weighting for display parameter contributions consistent with the integrated display values  $w_{ki}$  and  $\{\beta_j X_j\}$ .

7. Item 6 proposes the utility of the matrix and optimization technique as an in-process design and on-line data analysis tool for selection of the distribution of display contributions for each equipment or data parameter and, hence, selection of the array of signal levels for parameter central values.

#### IV. RECOMMENDATIONS.

1. The observation matrix technique for test data display and analysis be subjected to additional study as a basis for on-line operator decision making.

2. The matrices be stored in the same form as used for real-time data display as a part of a historical data bank.

3. Additional design and utility studies be directed to the suggested techniques as a basis for ADAPT and competitive DMS.



## REFERENCES

Gamble, Edward H., "Systems Modeling via Functions," Technical Report CS-A-74-1, May 1974.

Gamble, Edward H., "Design Analysis of Large-Scale Systems," Proceedings of the 1975 Army Numerical Analysis and Computers Conference, ARO Report 75-3, December 1975.

Gamble, Edward H., "Utilization of Computer Analyses for Major Design Decisions for Large-Scale Systems," Proceedings of the 1975 Army Numerical Analysis and Computers Conference, ARO Report 75-3, December 1975.

Gamble, Edward H., "A Function Modeling Technique for Estimating Computer Traffic for a Data Management System," DA 75-3, 15 July 1975.

Gamble, Edward H., "A Critique on the Design of Complex Systems," Copyright #A927900, 1967.



LETHALITY OF A SPECTRUM OF SHAPED CHARGE PROJECTILE  
ANTI-TANK FIREPOWER-KILL EFFECTS EVALUATED BY  
THE AVVAM-1 COMPUTER MODEL

Donald F. Haskell  
US Army Ballistic Research Laboratories  
Aberdeen Proving Ground, Maryland 21005

**ABSTRACT.** This paper describes an anti-tank lethality study performed by the use of AVVAM-1, the first version of a new armored vehicle vulnerability analysis computer model developed at the Ballistic Research Laboratories. In this study the vulnerability of a Russian T55 tank to a spectrum of anti-tank shaped charge projectile terminal ballistic effects is calculated and analyzed. This is done to illuminate the behind-armor effects characteristics and their combinations that are most lethal. The measure of lethality employed in this study is the probability of achieving a firepower kill given a hit on a specific point on the tank.

Lethality of the various projectile effects is calculated for direct frontal attack of the tank. It is assumed that there is an equal probability of hitting any point on the vehicle attack aspect. In this manner the individual projectile effects may be separated from weapons systems hit probability. Typical shaped charge projectile spatial distributions are employed for the mass, speed and numbers of fragments produced behind the armor. The end product of the study is a ranking of the damage producing effects of the various projectile types according to their lethality in causing a firepower kill. This was obtained by a sensitivity study of the effects of the various parameters. The conclusions of the study are:

1. Over the range of the variables (normalized to the standard BRL 3.3 inch cone diameter precision shaped charge) from 0.6 to 1.2, firepower-kill lethality of the shaped charge jet projectile in direct frontal attack of the Russian T55 tank is most sensitive to changes in cone diameter and the number and speed of behind-the-armor fragments (given in decreasing order of influence on lethality of the seven variables studied in this investigation).

2. Over the range of the standard charge normalized variables between 1.2 and 1.8, lethality is most sensitive to changes in jet velocity, cone diameter and the number of behind-the-armor fragments (given in decreasing order of influence). Here again, lethality is least sensitive to jet breakup time.

3. On the other hand, jet breakup time exerts the highest effect on lethality over the standard charge normalized variable range from 1.8 to 2.0. This is followed by jet velocity and cone diameter. Mass of the behind-the-armor fragments has the least influence on shaped charge lethality over this range.

4. A marked functional dependence of  $P_{K/H}$  upon the behind-the-armor fragment spray total kinetic energy was found. This information should be very useful in the design of new shaped-charge anti-tank projectiles. It should also provide guidance in the vulnerability reduction of tanks.

1. INTRODUCTION. The object of this investigation was to determine which of seven selected shaped charge munition parameters exert the most influence on the probability of achieving a firepower kill of the Russian T55 tank given a random hit on the tank in a direct frontal attack. The seven parameters are: the liner cone base diameter, munition standoff distance from the target, jet velocity, and jet break-up time as well as the number, mass and speed of the fragments generated behind the tank armor by the jet. Each parameter was varied independently over a pre-selected range of values to assess its effect on the firepower capability of the tank. In addition, the effects on kill probability caused by variation of combinations of these parameters was also studied.

AVVAM-1<sup>1</sup> was used to perform the study. AVVAM-1 (Armored Vehicle Vulnerability Analysis Model, first version) is a conceptual model and associated digital computer code developed at BRL to analytically assess the vulnerability of armored vehicles. AVVAM-1 can be employed to perform both armored vehicle vulnerability and anti-armor weapons design and analysis studies. This first version of AVVAM treats components and personnel subjected to penetration and/or perforation damage mechanisms. The attacking munition may be a shaped charge or kinetic energy projectile, or a shaped charge or Misznay-Schardin land mine. With additional effort the present model may be extended to include other damage mechanisms. Although originally developed for armored vehicles, the code is not restricted to armored vehicles - it may be employed to assess the vulnerability of any structure.

AVVAM-1 is an outgrowth of an existing digital computer code developed by H. Ege of the Surface Targets Branch, Vulnerability Laboratory, BRL. Ege's code is based on relations between the characteristics of certain weapons and vehicle damage observed from the results of antitank tests conducted under the auspices of the UK, Canada and the US in Canada during 1959.<sup>2</sup>

<sup>1</sup>D. F. Haskell and M. J. Reisinger, "Armored Vehicle Vulnerability Analysis Model - First Version," US Army Ballistic Research Laboratories Interim Memorandum Report No. 85, February 1973.

<sup>2</sup>Canadian Armament Research and Development Establishment Report Q-21, "Tripartite Anti-Tank Trials and Lethality Evaluation (U)," Final Report Part I, November 1959, (SECRET).

AVVAM-1 is based on analytical evaluations of the damage inflicted on individual critical components and the aggregate effect of these damaged components on compartment and overall vehicle vulnerability. To do this, AVVAM-1 accounts for not only the damage inflicted on components in the direct line of fire, or shotline, of the attacking munition but also the damage inflicted by armor spall and/or munition fragment sprays on components located away from the munition shotline. In addition, AVVAM-1 accounts for the degrading (or possible enhancing) effects on the spall and/or fragment sprays caused by components positioned between the armor and the critical components. Thus, the potential protection afforded critical components by intervening components is included in the AVVAM-1 calculational procedure.

2. DESCRIPTION OF AVVAM-1. AVVAM-1 is composed of two major computer codes. One of these characterizes the target. The other code characterizes the munition-target interaction and performs the vulnerability evaluation. The target characterization code describes the target and identifies, locates and determines the presented area of critical components. It also provides information concerning components that are located between the vehicle armor and the critical components.

To generate the target description information, AVVAM-1 employs the GIFT (Geometric Information for a Target) code.<sup>3</sup> The GIFT code is an improved version of the existing MAGIC code.<sup>4</sup> The identification, location and presented area determinations of critical components and the intervening component information is generated by a subcode within the GIFT code called RIP (Rays Initiated at a Point).

The second major code employed in AVVAM-1 encompasses the terminal ballistics of the attacking munition and the post-plate-perforation characteristics of plate spall or munition fragment sprays. In addition, this second code calculates the vulnerability of selected components within the vehicle as well as compartment vulnerability and overall vehicle vulnerability. Because of its functions, it is called the P<sup>3</sup> and C<sup>3</sup>PKH (Post-Plate-Perforation and Component, Compartment and Combat Vehicle Probability of a Kill given a Hit) code.

In operation, AVVAM-1 selects critical components within the target and then evaluates the extent of damage and kill probability for each selected munition aim point in a given view of the target. It does this

---

<sup>3</sup> Lawrence W. Bain, Jr., and Mathew J. Reisinger, "The GIFT Code User Manual, Volume I Introduction and Input Requirements," BRL Report No. 1802, July 1975.

<sup>4</sup> Armament Systems, Incorporated and Propulsion Development Department, "MAGIC Computer Simulation," Volumes I and II, Naval Weapons Center Technical Note 4565-3-71, Volume I and Volume II, May 1971.

by determining the armor thickness in the direction of the shotline of the attacking munition and the number of intervening components between the vehicle armor and critical component. It then utilizes the behind-the-plate characterization of a specific munition to calculate kill probabilities given a hit for all or selected critical components within the vehicle. This whole process is accomplished by firing a large number of parallel rays at a given attack angle and azimuth into the target. Each individual parallel ray then spawns new rays that are initiated at the munition exit point on the armor interior surface. These new rays are used to search out the critical components, define their position, shielding, and presented area. Then the post-plate-perforation subcode converts terminal ballistics input data into an expected number of hits into each of the critical components and finally the  $C^3PKH$  subcode determines the probability of a kill of these components for the expected number of hits. The kill probabilities for all the critical components within a given compartment may be combined into compartment mobility (M), firepower (F), and complete or catastrophic (K) kills. Values for M, F, and K kills of the whole vehicle may also be determined.

A flow chart summarizing the operations of AVVAM-1 is presented by Figure 1. In this figure Box 1 represents the target input. Box 2 is the RIP section of the GIFT code, Box 3 is the  $P^3$  section and Box 4 is the  $C^3PKH$  section. The  $C^3PKH$  section provides the output in terms of probability of a kill given a hit. Also indicated in the figure is Box 5 which indicates an iteration scheme that may be employed for multiple views. Since the sections represented by Boxes 1, 2, 3, and 4 provide the PK/H output for a single view, results for multiple views may be obtained by iterating through Boxes 2, 3, and 4 for each view desired.

The code operates as follows: The particular target description is input through Box 1 on cards and the specific munition is input by cards through Box 3. Information for the  $P^3$  section is handled by card input. After the target is described and the critical components identified, for a single vehicle view, RIP selects a starting point on the vehicle, fires a main ray at the starting point, and essentially determines the position, shielding, and presented area of all the critical components in the vehicle in relation to the shotline of the main ray. The  $C^3PKH$  code calls on the Post-Plate-Perforation code to supply the behind the plate spall data and main munition shotline information to include number of fragments, size, and speed of fragments. Next, it calculates the expected number of fragments to hit a given critical component, and then the probability of killing that component given a hit. It does this for each critical component identified by the RIP code for the particular shotline selected. All the critical components are evaluated for the first shotline. The RIP code then moves to a new shotline (or shotpoint)

and the probabilities of a kill given a hit are calculated for all the components in the view of the new shotline. This process is continued until the whole view of the vehicle is completed. At this point the output of the AVVAM code is the following: Probability of a kill for each critical component in the vehicle, a set of compartment M, F, and K kill probabilities and overall vehicle view probability of M, F, and K kill values. During these calculations the  $C^3PKH$  code in conjunction with the  $P^3$  code account for the mass and velocity attrition of the shotline and spall fragments as they perforate intervening components between the exit point on the armor and the specific critical component under evaluation at that time.

3. STUDY CONDITIONS. As described previously, the object of this investigation was to determine which of the following parameters has the most effect on the firepower kill lethality of a shaped charge projectile used in a direct frontal attack of the T55 tank: liner cone base diameter, standoff distance, jet velocity, jet breakup time, and the number, mass and speed of the behind-the-armor fragments produced by the jet. Each of these parameters was varied independently over a selected range of values and AVVAM-1 was used to calculate and follow the attendant variation in firepower kill probability given a hit on the tank ( $P_{K/H}$ ). The effect on  $P_{K/H}$  of certain combinations of parameters was also calculated. To do this, equal variations in each of the parameters in the combination were employed.

Those parameters used to characterize the shaped charge jet itself were varied over the following range of values:

- . Liner cone base diameter = 0.0508 m -0.2032 m
- . Munition-target standoff distance = 0.08382 m -2.0955 m
- . Jet velocity = 2,000 m/sec -20,000 m/sec
- . Jet break-up time = 25 microsecs -200 microsecs

The behind-the-armor fragment characteristics produced by the above range of conditions were automatically calculated internally by AVVAM-1. In addition, other calculations were made in which the behind-the-armor fragment characteristics (number of fragments, fragment mass and fragment speed) were varied independently of the normally expected values that would otherwise be governed by the jet parameters. In these cases, the number of fragments, their mass and speed were made proportional to those produced by a standard shaped charge and were varied over the range from 0.01 to 10 times the particular standard charge behind-the-armor characteristic value. That is, the number of fragments-to-number of fragments from the standard charge ratio, fragment mass-to-standard charge fragment

mass ratio, and the fragment speed-to-standard charge fragment speed ratio were varied from 0.01 to 10.

The standard shaped charge<sup>5</sup> is a 42°, apex angle, copper cone loaded with octol explosive in a light confining wall of aluminum. A diagram of this charge is shown in Figure 2. All linear dimensions in the diagram are in inch units. It will be referred to as the "standard charge." The nominal liner wall thickness is .081 inch (.206 mm) and the outside diameter at the base of the cone is 3.3 inches (84 mm). The total liner weight is .61 lb (.277 kg), the explosive weight is 1.93 lbs (.875 kg), and the aluminum body weight is 1.14 lbs (.517 kg). The standard cone diameter and standoff as well as the jet constants of this standard charge obtained experimentally from flash radiographs are as follows:

Cone diameter = 0.0832 m  
Standoff = 0.16764 m  
Jet velocity = 8,300 m/sec  
Break-up time = 103 microseconds

The T55 tank was assumed to be fully combat loaded and directly attacked in the front (zero azimuth and elevation) by the shaped charge projectile. A planar gridwork of 60.96 cm square cells was erected over the front of the tank normal to the attack direction. A shotline, to simulate a projectile flight path, was fired in the attack direction at a random point within each cell. In this manner the projectile lethality, measured by the firepower  $P_{K/H}$ , was evaluated for strike points over the whole front of the tank. Then these individual cell (or shotline)  $P_{K/H}$ 's were averaged together to obtain a single firepower kill  $P_{K/H}$ , representative of the particular set of parameter values being evaluated.

The GIFT description of the T55 tank consisted of approximately 630 components. Of this total about 400 were considered critical components. About half of these were considered critical to the tank's firepower. In this AVVAM-1 analysis, 5 rays to simulate behind-the-armor fragments were fired at each of these 200 firepower critical components.

4. RESULTS AND DISCUSSION. The study results obtained by use of AVVAM-1 are illustrated in Figures 3 through 14 and Tables I and II. Figures 3 through 6 show the effects on average firepower kill  $P_{K/H}$  of independent variations in the basic shaped charge projectile jet characterization parameters. In these figures the ordinate corresponds to  $P_{K/H}$  and the abscissa corresponds to the jet parameter. The jet parameters are

<sup>5</sup>R. DiPersio, J. Simon and A. B. Merendino, "Penetration of Shaped-Charge Jets Into Metallic Targets," BRL Report No. 1296, September 1965, pp. 16-17.



displayed as a ratio of the actual parameter value-to-the value of that parameter exhibited by the "standard charge." For example, in Figure 3, at the abscissa value equal to 2, the cone diameter under investigation is twice the size of the standard charge cone diameter, or .0168 m (6.6 in.). Each parameter was varied while the remaining parameters were kept constant at the value exhibited by the standard charge. In the cone diameter case the standoff was maintained equal to that of the standard charge i.e., equal to two cone diameters.

As to be expected, the figures show that average  $P_{K/H}$  increases with increasing cone diameter, jet velocity and breakup time and that  $P_{K/H}$  decreases as standoff is increased. The  $P_{K/H}$  and its slope vary continuously with cone diameter. On the other hand, both  $P_{K/H}$  and its slope vary irregularly (almost discontinuously) with jet velocity and breakup time. Figure 5 depicts a plateau in  $P_{K/H}$  over the jet velocity-to-standard charge jet velocity ratio from 0.6 to 1.3. A wider plateau exists in the relationship between  $P_{K/H}$  and breakup time as shown by Figure 6. In this case the plateau extends from breakup time-to-standard charge breakup time ratio equal to 0.4 to approximately 1.6.

The relationship between  $P_{K/H}$  and standoff illustrated in Figure 4 is interesting. As to be expected,  $P_{K/H}$  decreases with standoff over the range studied. However, this decrease is less than would be expected-the functional dependence of  $P_{K/H}$  on standoff is quite weak. It is much less than it is with the other jet parameters. The jet lethality is not significantly degraded at standoffs much greater than standard.

Figure 7 illustrates the effect on  $P_{K/H}$  of simultaneous and equal variation of the jet parameters. As before, the abscissa represents the value of the parameter relative to its value exhibited by the standard charge. Although here it represents the value of all four jet parameters relative to the standard charge case. For example, at the parameter value-to-parameter value of standard charge ratio equal to 0.5, the  $P_{K/H}$  computation was performed with the cone diameter, standoff, jet velocity and jet breakup time all set equal to one-half the parameter values corresponding to the standard charge. As indicated by the figure,  $P_{K/H}$  is a smooth, increasing function of the four jet parameters. Its rate of increase with these four parameters is higher, as to be expected, than its variation with change in the parameters individually, although the combined effect is not the sum of the effects of the individual parameters.

Figure 8 shows the effect of individual and combined change in the behind-the-armor fragment parameters on  $P_{K/H}$ . The functional relationship between  $P_{K/H}$  and the fragment parameters is similar for all three: number of fragments, fragment mass and speed.  $P_{K/H}$  increases rapidly as the fragment parameter-to-standard charge parameter value ratio increases from zero to 1 or 2. Beyond this region, the  $P_{K/H}$  rise levels off. As indicated, equal variation of all three fragment parameters at the same time yields higher  $P_{K/H}$  than individual variation of these parameters. However, the combined effects do not show as high an increase over the individual fragment parameter variations as may be expected beforehand.

The slopes of the curves from Figures 3 to 6 and 8 in which the jet and fragment parameters are varied independently are plotted in Figures 9 and 10. Figures 9 and 10 exhibit the  $P_{K/H}$  sensitivity to the various parameters over a range of these parameters. As indicated, there is no single parameter that has the most influence on  $P_{K/H}$  over the range of independent parameters shown. Between normalized variable from 0.6 to about 1.2,  $P_{K/H}$  is most sensitive to cone diameter. Above 1.2 and below about 1.8 jet velocity has the most effect on  $P_{K/H}$ . Between normalized variable equal to 1.8 and 2.0, the jet velocity displays the highest influence on  $P_{K/H}$ . The effect of standoff, as described earlier, is minimal over the whole range. In regard to the fragment effects, the number and speed of the fragments show about the same influence on  $P_{K/H}$  which, over most of parameter range shown, is considerably higher than the fragment mass influence.

The slopes of the curves in Figures 9 and 10 over the range of parameter-to-standard charge ratio from 0.6 to 2.0 at discrete points within this range are listed in Table I. The slopes are normalized to that of the cone diameter. In this manner those parameters that have higher, or lower, influence on  $P_{K/H}$  than the cone diameter are easily distinguished. Table II lists the parameters according to their decreasing order of influence on  $P_{K/H}$  over the variable value normalized to standard charge range from .6 to 2. As indicated, cone diameter, jet velocity and jet breakup time exhibit the highest influence on  $P_{K/H}$  over the range of variables shown.

Figures 11 to 14 are log-log plots of  $P_{K/H}$  versus behind-the-armor fragment total kinetic energy. These figures are included here to illustrate the relationship between behind-the-armor fragment kinetic energy

and average kill probability. Figures 11, 12 and 13 show the calculation results for independent variations in the fragment parameters. In Figure 11 the total fragment kinetic energy was increased by increasing the number of fragments with their mass and speed held constant and equal to the mass and speed exhibited by the standard charge. In Figure 12 the mass of the fragments was varied while the number and speed were maintained equal to those characteristic of the standard charge. The fragment speed was varied in Figure 13, and in Figure 14 all three parameters were varied simultaneously and equally. The lines in these figures were drawn in by "eye." These figures indicate that there appears to be a definite relationship between behind-the-armor fragment total kinetic energy and resultant  $P_{K/H}$ . In each figure,  $P_{K/H}$  increases with total fragment kinetic energy. Furthermore, there are two distinct regions of fragment influence, with kinetic energy of the order of  $10^4$  to  $10^5$  joules as the demarcation zone between these two regions. Fragment kinetic energy has a much larger effect on  $P_{K/H}$  in the lower kinetic energy region than in the higher region.

## 5. CONCLUSIONS.

1. Over the range of the variables (normalized to the standard BRL 3.3 inch cone diameter precision shaped charge) from 0.6 to 1.2, firepower-kill lethality of the shaped charge jet projectile in direct frontal attack of the Russian T55 tank is most sensitive to changes in cone diameter and the number and speed of behind-the-armor fragments (given in decreasing order of influence). Over this same range, jet breakup time exerts the least influence on lethality of the seven variables studied in this investigation.
2. Over the range of the standard charge normalized variables between 1.2 and 1.8, lethality is most sensitive to changes in jet velocity, cone diameter and the number of behind-the-armor fragments (given in decreasing order of influence). Here again, lethality is least sensitive to jet breakup time.
3. On the other hand, jet breakup time exerts the highest effect on lethality over the standard charge normalized variable range from 1.8 to 2.0. This is followed by jet velocity and cone diameter. Mass of the behind-the-armor fragments has the least influence on shaped charge lethality over this range.
4. A marked functional dependence of  $P_{K/H}$  upon the behind-the-armor fragment spray total kinetic energy was found.

#### REFERENCES

1. D. F. Haskell and M. J. Reisinger, "Armored Vehicle Vulnerability Analysis Model - First Version," US Army Ballistic Research Laboratories Interim Memorandum Report No. 85, February 1973.
2. Canadian Armament Research and Development Establishment Report Q-21, "Tripartite Anti-Tank Trials and Lethality Evaluation (U)," Final Report Part I, November 1959, (SECRET).
3. Lawrence W. Bain, Jr., and Mathew J. Reisinger, "The GIFT Code User Manual, Volume I Introduction and Input Requirements," BRL Report No. 1802, July 1975.
4. Armament Systems, Incorporated and Propulsion Development Department, "MAGIC Computer Simulation" Volumes I and II, Naval Weapons Center Technical Note 4565-3-71, Volume I and Volume II, May 1971.
5. R. DiPersio, J. Simon and A. B. Merendino, "Penetration of Shaped-Charge Jets Into Metallic Targets," BRL Report No. 1296, Sep 65.

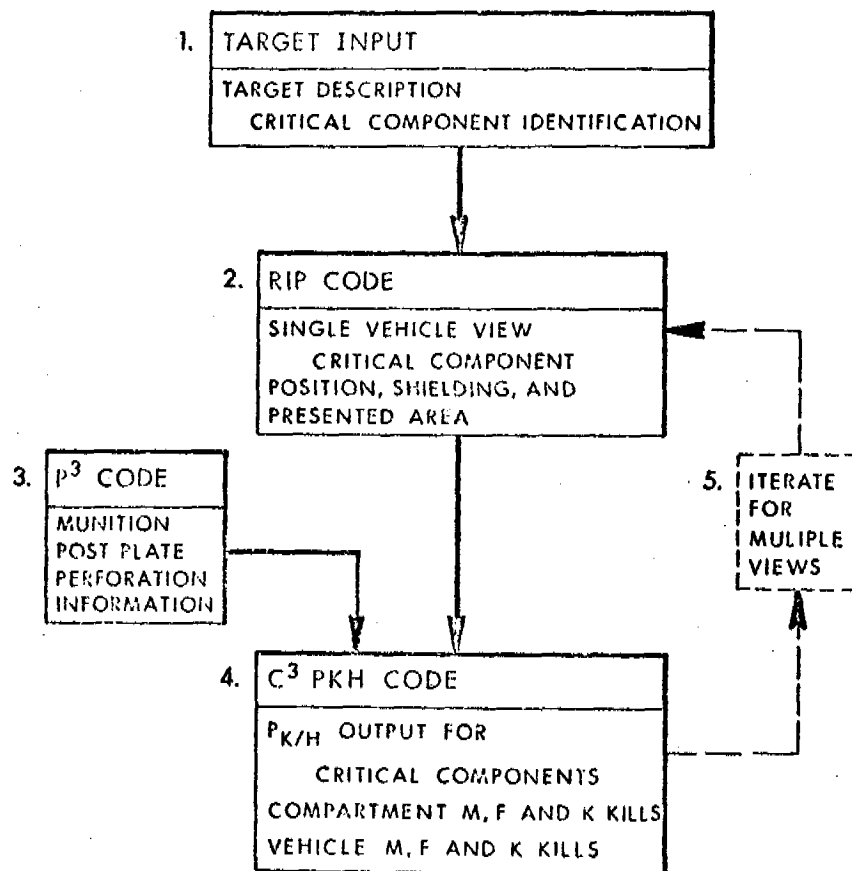


Figure 1. AVVAM-1 Code Summary Flow Chart

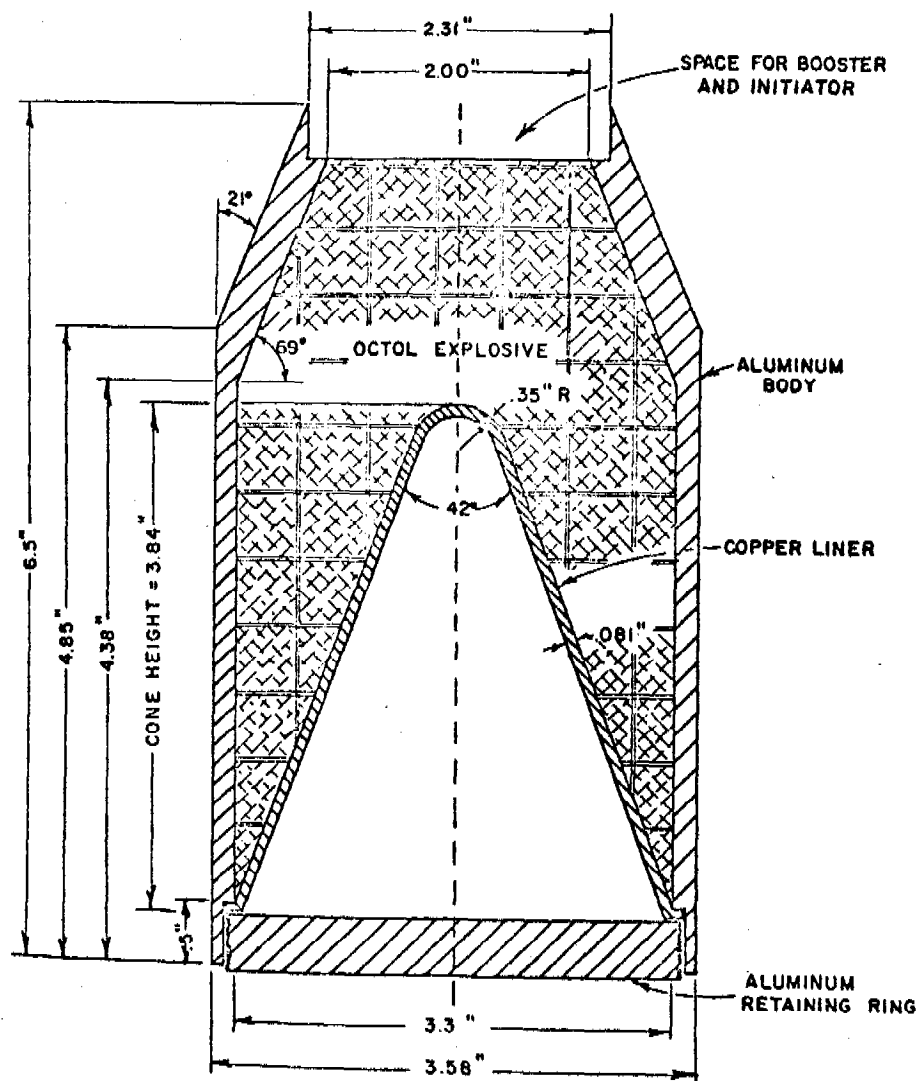


Figure 2. Diagram of Standard Shaped Charge

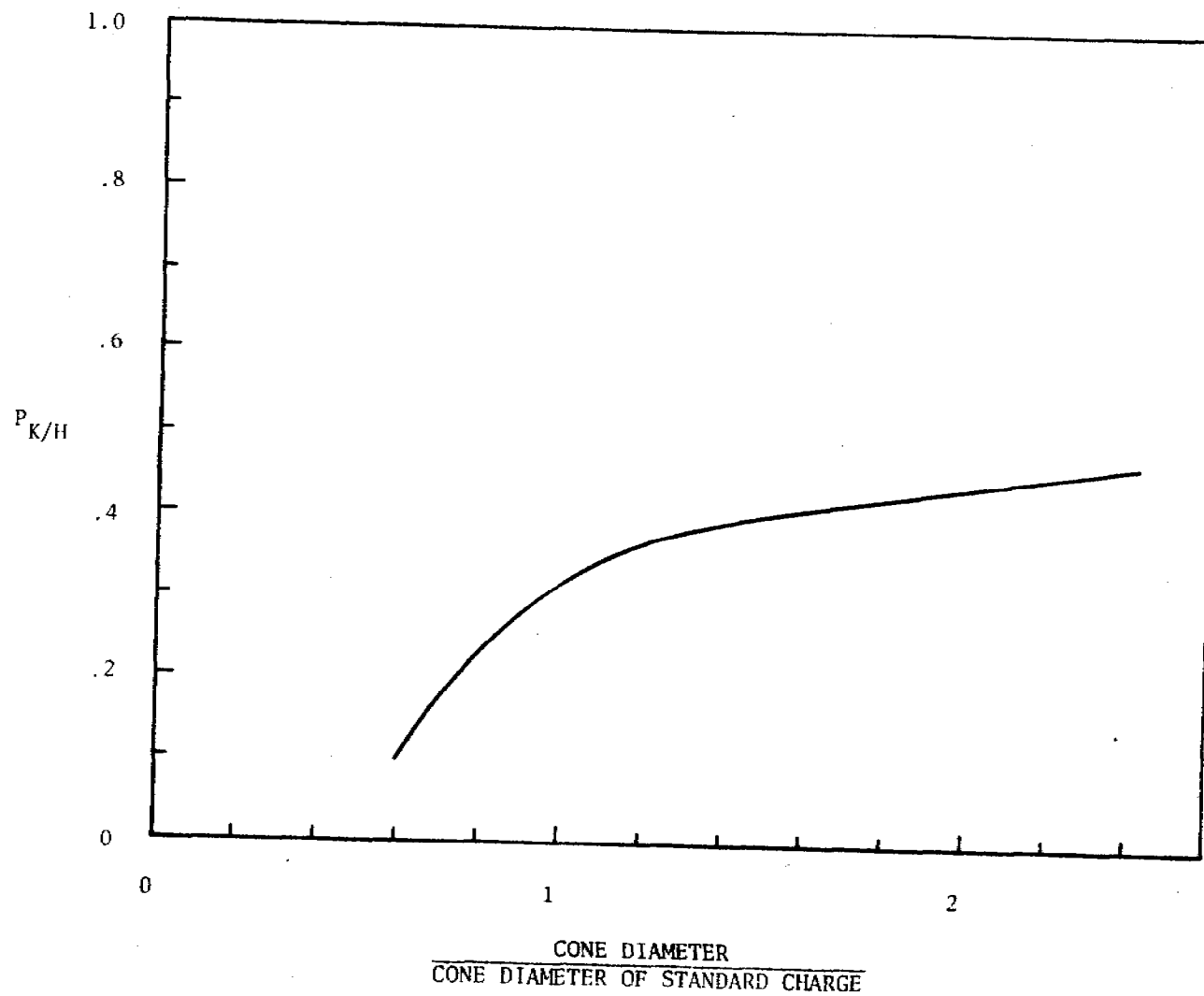


Figure 3. Effect of Liner Cone Diameter

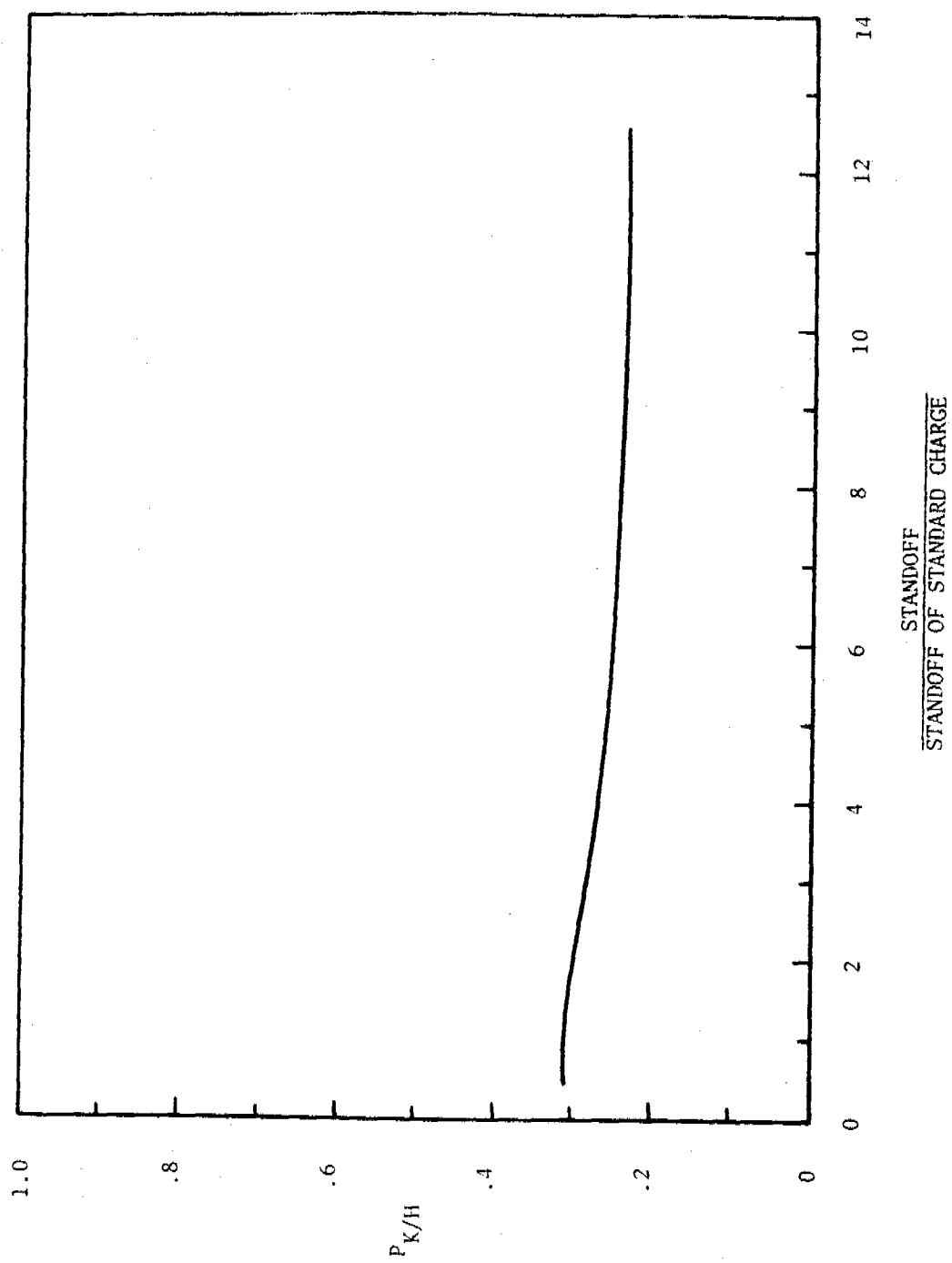


Figure 4. Effect of Charge Standoff



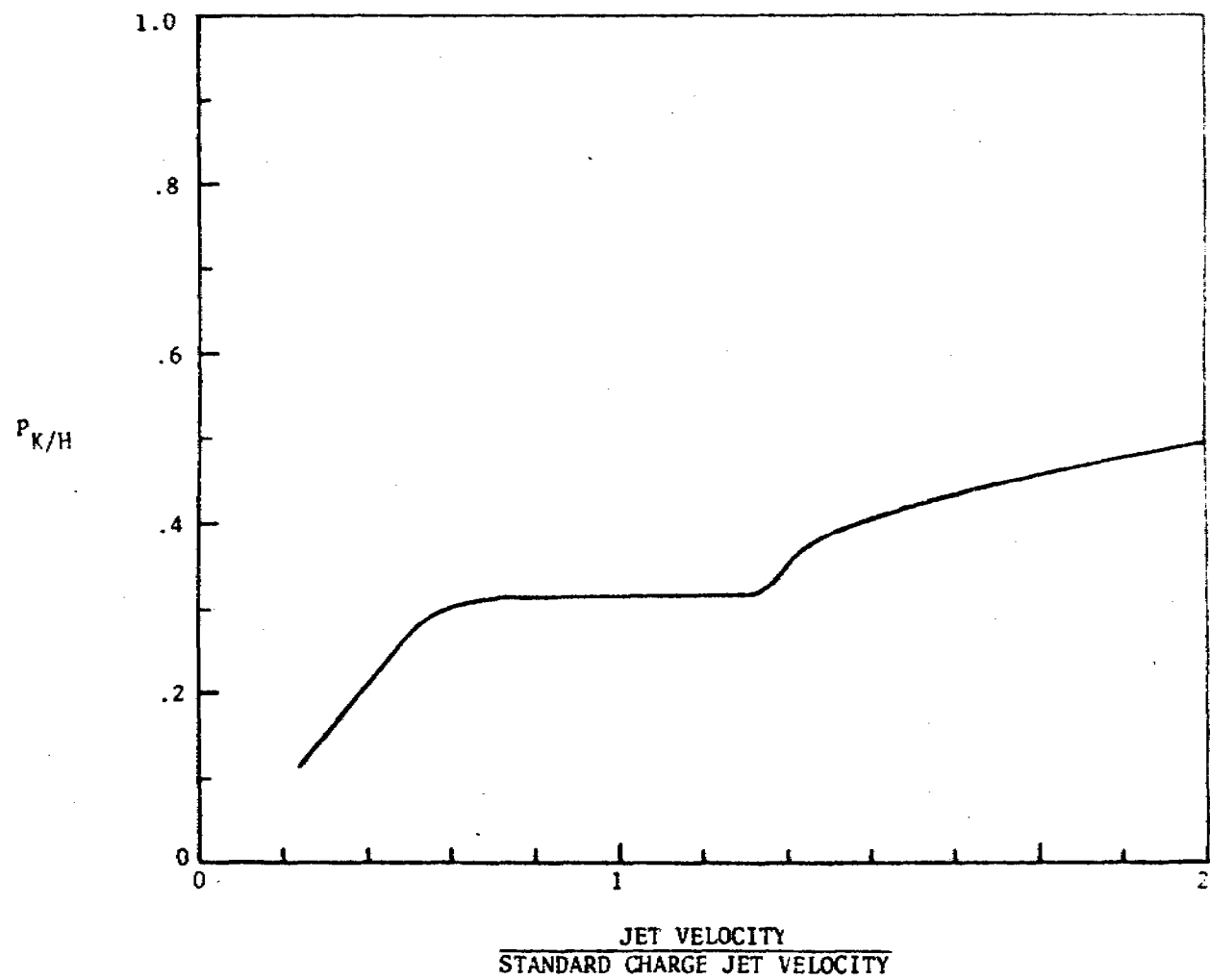


Figure 5. Jet Velocity Influence

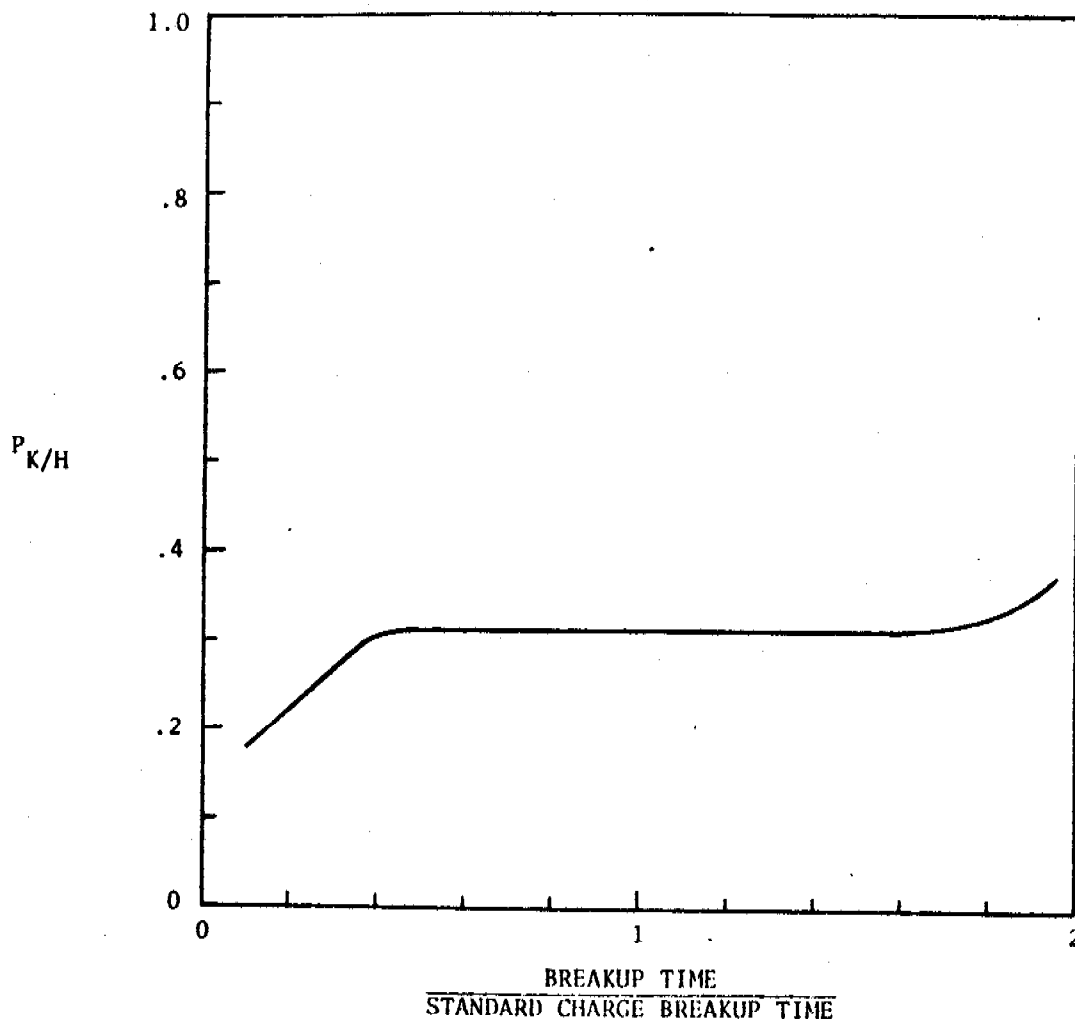


Figure 6. Jet Breakup Time Influence

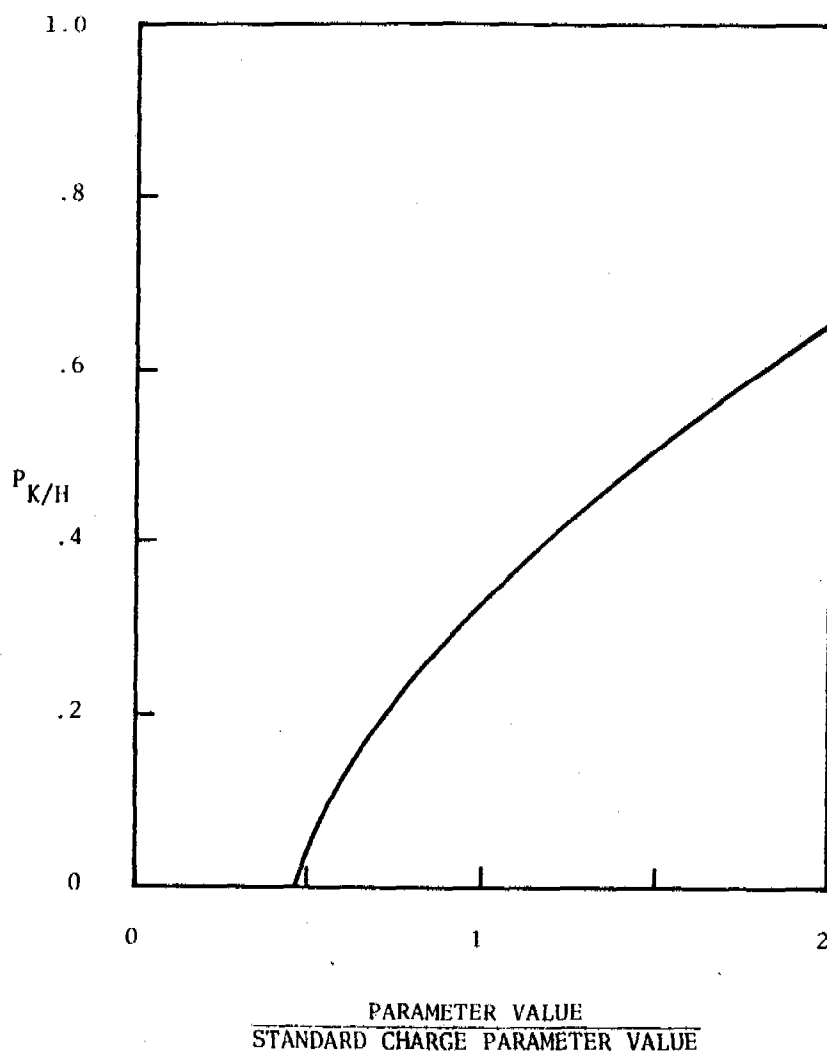


Figure 7. Effect of Simultaneous and Equal Variation in Cone Diameter, Standoff, Jet Velocity and Jet Breakup Time on  $P_{K/H}$

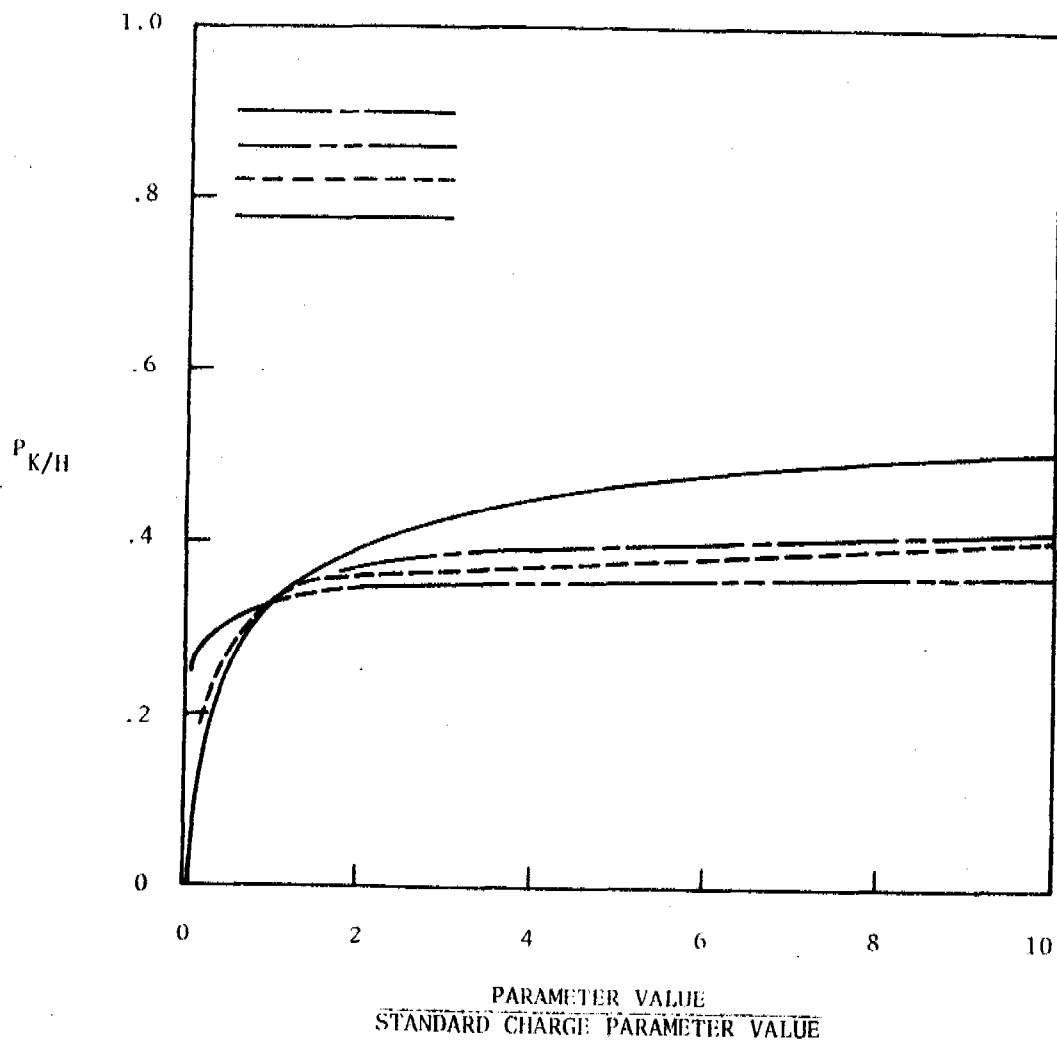


Figure 8. Effect of Behind-the-Armor Fragment Parameters on  $P_{K/H}$

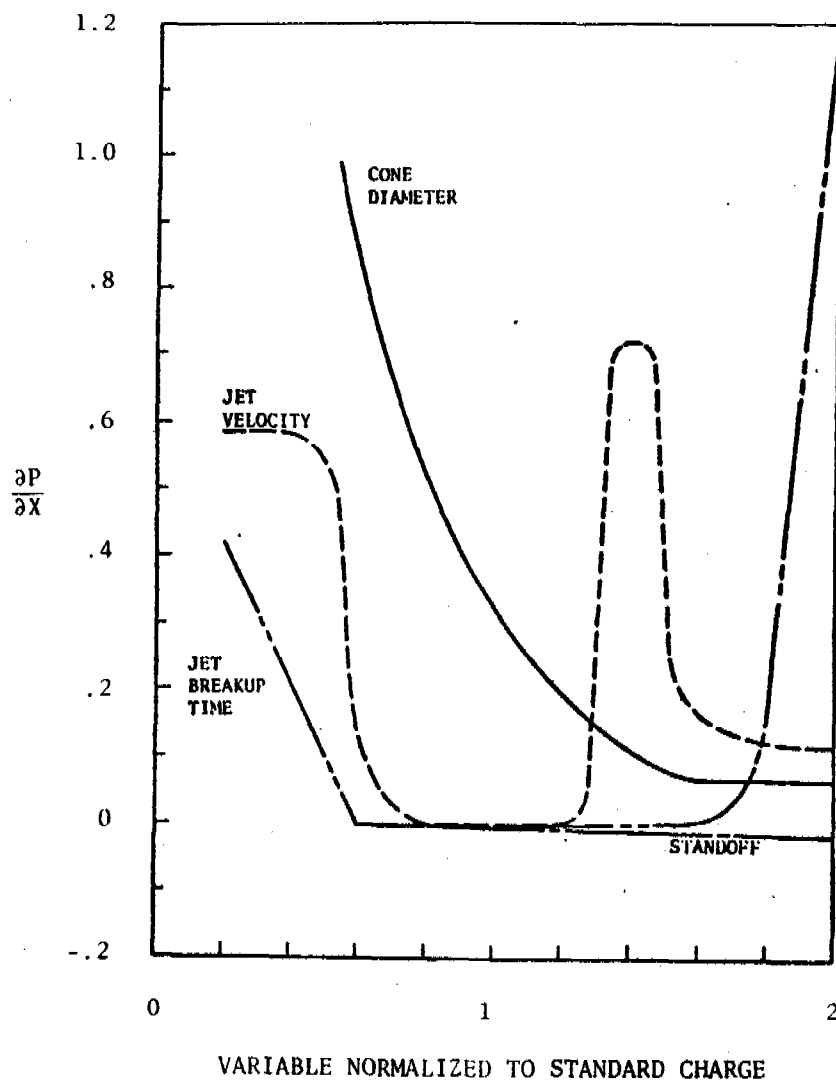


Figure 9. Sensitivity to Shaped Charge Variables

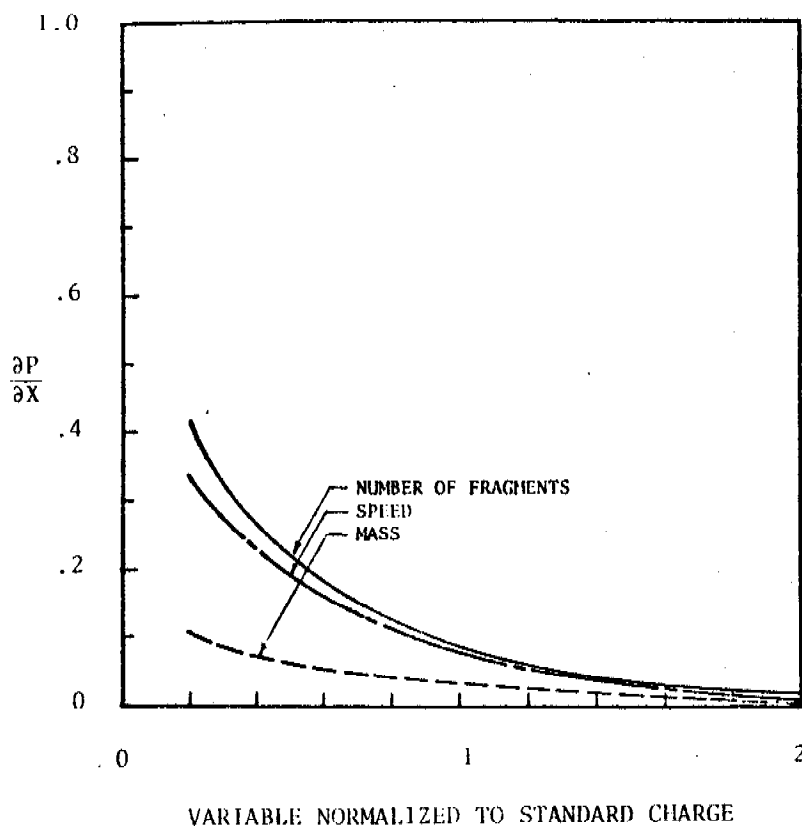


Figure 10. Sensitivity to Fragment Variables

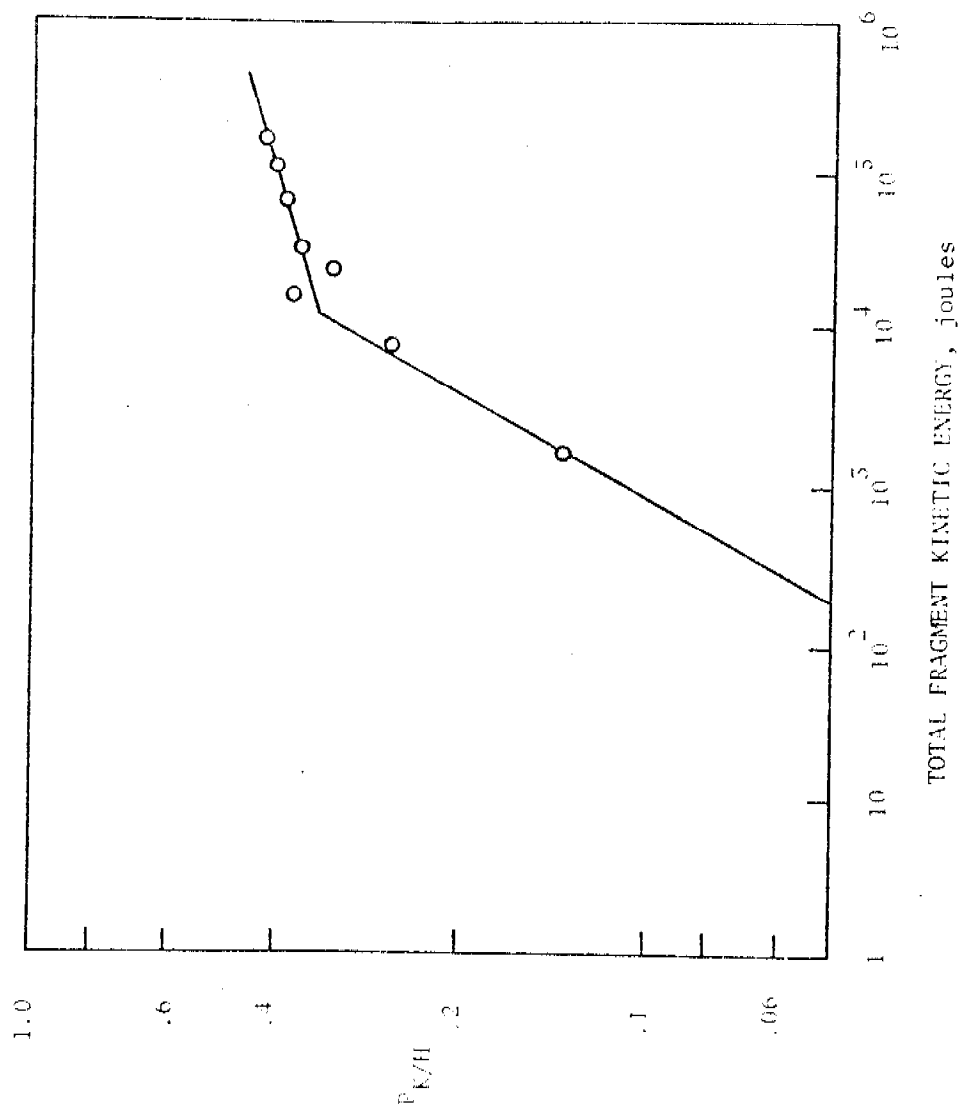


Figure 11.  $P_K/H$  Variation with Total Fragment Kinetic Energy Produced by Variation in Number of Fragments

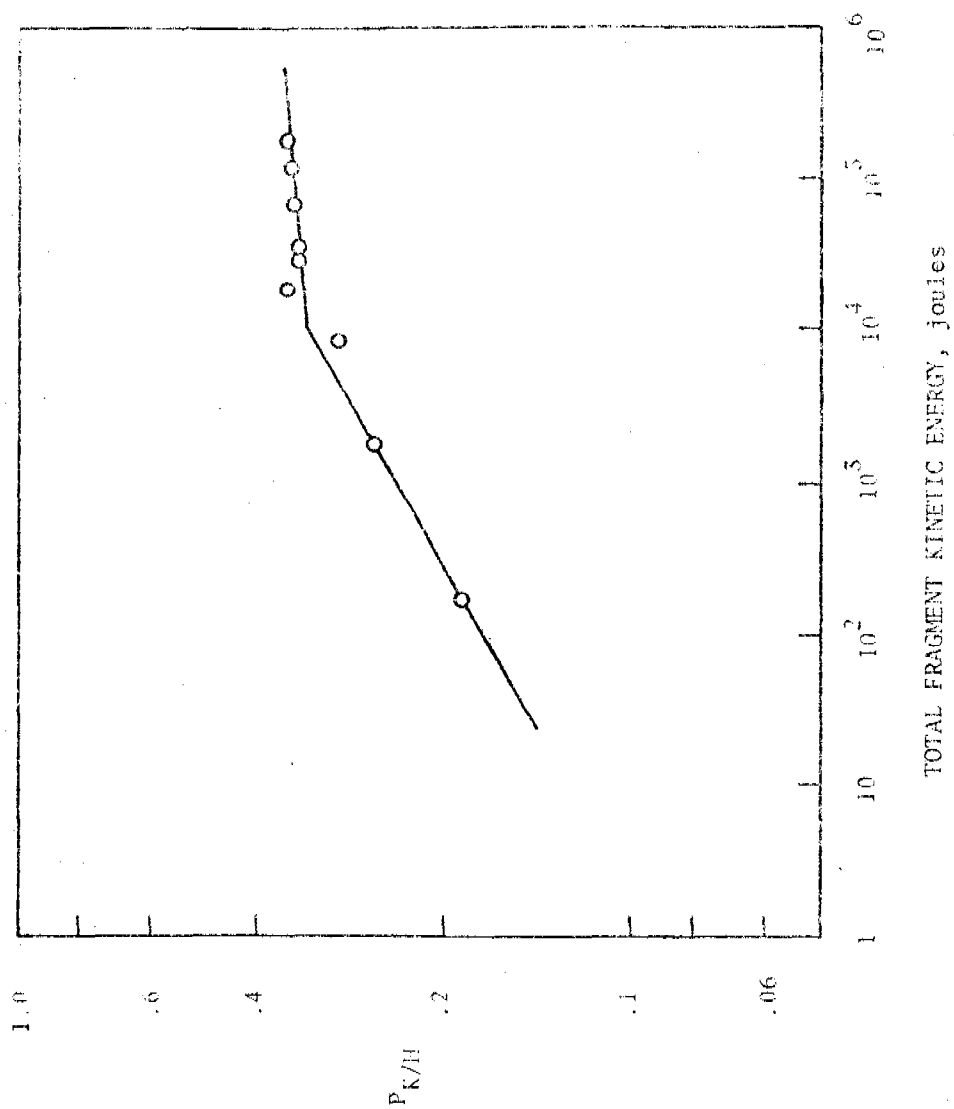


Figure 12.  $P_{K/H}$  Variation with Total Fragment Kinetic Energy Produced by Variation in Fragment Mass



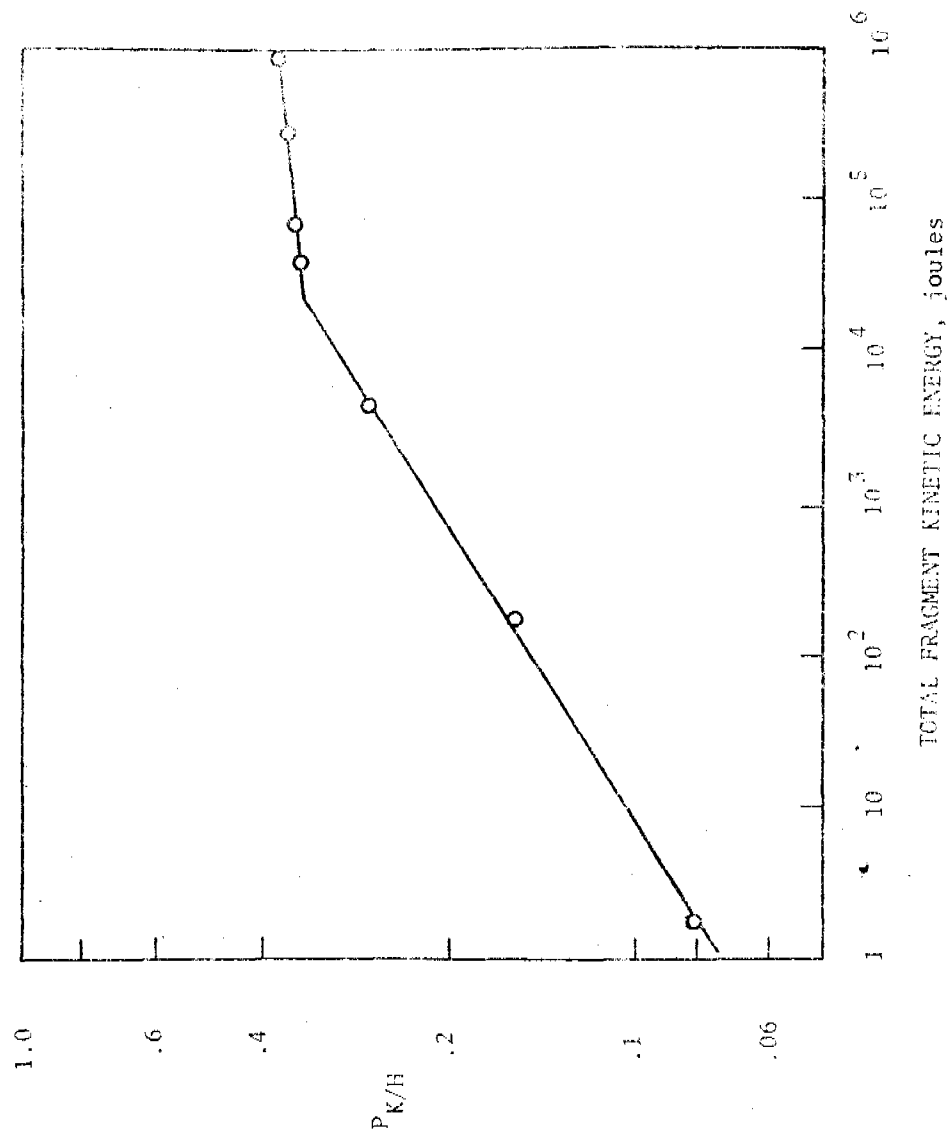


Figure 13.  $P_{K/H}$  Variation with Total Fragment Kinetic Energy Produced By Variation in Fragment Speed

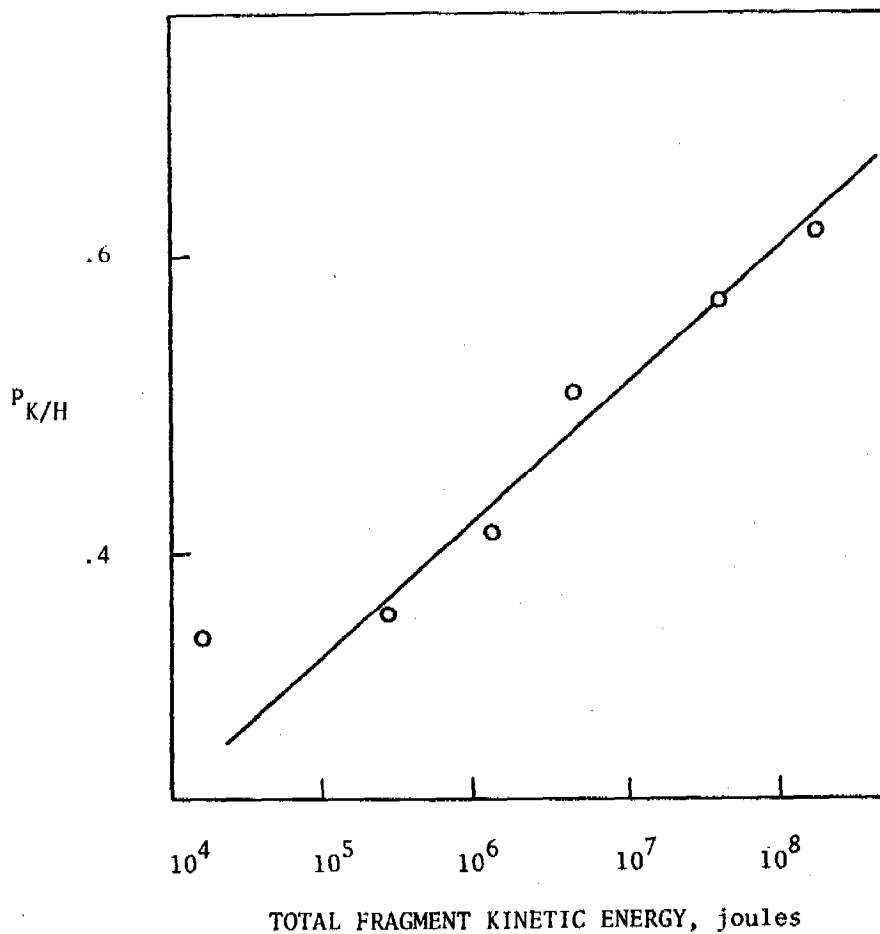


Figure 14.  $P_{K/H}$  Variation with Total Fragment Kinetic Energy for Simultaneous and Equal Variations in Fragment Number, Mass and Speed

Table I.  $P_{K/II}$  Sensitivities Relative to Cone Diameter

VARIABLE	.6	.8	1	1.2	1.4	1.6	1.8	2.0
CONE DIA.	1	1	1	1	1	1	1	1
STANDOFF	.005	-.008	-.024	-.057	-.079	-.174	-.203	-.217
JET VELOCITY	.172	0	0	0	6.263	2.333	1.855	1.681
BREAKUP TIME	0	0	0	0	0	0	2.203	14.493
NUMBER OF FRAGS.	.211	.236	.246	.359	.351	.435	.319	.304
MASS OF FRAGS.	.072	.098	.114	.156	.316	.261	.145	.058
FRAGMENT SPEED	.187	.226	.237	.286	.202	.319	.261	.116

Table II.  $P_{K/H}$  Sensitivities in Descending Order of Influence

INFLUENCE	VARIABLE VALUE NORMALIZED TO STANDARD CHARGE							
	.6	.8	1	1.2	1.4	1.6	1.8	2
HIGHEST	CD	CD	CD	CD	VJET	VJET	T1	T1
	NF	NF	NF	NF	CD	CD	VJET	VJET
	SF	SF	SF	SF	NF	NF	CD	CD
	VJET	MF	MF	MF	MF	SF	NF	NF
	MF	S	S	S	SF	MF	SF	S
	S	VJET	VJET	VJET	S	S	S	SF
LOWEST	T1	T1	T1	T1	T1	T1	MF	MF

NOTE:

CD: Cone diameter  
 VJET: Jet velocity  
 T1: Jet breakup time  
 S: Shaped charge standoff distance  
 NF: Number of behind-the-armor fragments  
 SF: Speed of behind-the-armor fragments  
 MF: Mass of behind-the-armor fragments

DEVELOPMENT AND OPTIMIZATION OF SIGNAL PROCESSING  
UTILIZED IN A MINE DETECTION SYSTEM

Abram Leff  
U.S. Army Mobility Equipment R&D Command  
Fort Belvoir, Virginia 22060

**ABSTRACT.** Based upon an extensive R&D program a broadband microwave technique has evolved which can be utilized to rapidly scan roads and reliably detect both metallic/non-metallic cased AT mines. In order to optimize mine detection capability and minimize false alarms, processing algorithms had to be developed and evaluated. Field data was initially recorded on analog tape for off-site processing. This data was utilized to develop potential processing algorithms.

To permit the evaluation of algorithms in real-world conditions, a mobile real-time feasibility system was designed and fabricated. This system incorporated a CDC 469 minicomputer which allowed storage in memory of up to six algorithms, as well as the control programs for the system. Flexibility incorporated in the design and the addition of a keyboard unit provided the capability to utilize any one of the six algorithms or vary most test parameters in the field, by simply changing an address in the System Control Unit or CPU.

This paper will discuss utilization of a minicomputer in the development of an optimized processing algorithm for mine detection.

1. **INTRODUCTION.** Considerable data processing and analysis effort has been expended with the objective of developing target discrimination techniques for detecting road mine responses in the presence of normal background responses. The goal of this effort was the development of field-operable software that optimizes the probability of detecting AT/AV mines under field conditions while concurrently minimizing the false alarm rate.

The inherent nature of the data analysis problem dictates a tradeoff between the probability of target detection vs. the false alarm rate. This relationship in a specific situation depends upon the particular field conditions encountered; therefore, large amounts of test data had to be gathered from a considerable variety of test site conditions in order to optimize detection vs. false alarms.

Initially, field data was recorded on a four channel analog tape recorder and was off-line converted to digital data format and recorded on IBM-compatible magnetic tape which then could be played back in various ways to produce plots of the desired data parameters. This digitized data also formed the basis for the development of the processing algorithms by means of computer analysis.

2. STATEMENT OF PROBLEM. To attack the basic data analysis problem, which was to develop a field-operable software system maximizing target detection while minimizing false alarms, a methodical investigation was initiated to (a) define the general target responses, (b) select the most favorable processing methods, (c) find the best way to represent the data, (d) identify the decision rules for discrimination between mine signature and background data, and (e) minimize the computational complexity needed to perform the task.

Based upon the results of the various analysis and processing schemes which were applied to test data, it was concluded that two initial processes were essential: (a) calculation of a background estimate based on preceding data inputs, and (b) normalization of the input data with respect to the background estimate. The latter is required because of the large differences in absolute amplitude levels in the various frequency channels caused by frequency roll-off of the antennas, and because of different attenuation levels in the soil.

3. ALGORITHM DEVELOPMENT. The first useful algorithm developed was named Target Amplitude Descriptor, or TAD, where for each channel and at each consecutive position the ratio between the input data and the background estimate was established, after which these ratios were averaged for all channels of a particular antenna pair.

Logarithmic TAD plots were made from the recorded test data using the amplitude information  $R_{ij}$  (ratio between receive and transmit signal levels in dB as measured at position  $i$  and frequency  $j$ ) in the following manner:

$$\text{Log TAD}_i = 20 \log \left\{ \frac{1}{n} \sum_{j=1}^{j=n} A_{ij}/B_{ij} \right\} \text{ (in dB)}$$

where:  $A_{ij}$  = linear value of  $R_{ij}$  (by using:  $A_{ij} = 10^{R_{ij}/20}$ )

$n$  = number of frequency channels

$B_{ij}$  = linear value of background estimate at position  $i$  and frequency  $j$ , determined as follows:

$$B_{ij} = (1 - \alpha) \times B_{i-1,j} + \alpha \times A_{i-1,j}$$

where  $\alpha$  is a constant chosen such that the background estimate relates to a selected distance:

$$\alpha = 1 - e^{\frac{-2\Delta X_i}{\text{Decay}}}$$

where  $\Delta X_i$  = position increment in meters

"Decay" can be defined also as the trailing distance where the relative weight of the background contribution is equal to 13.5%. This process is known as exponential smoothing (i.e. gives greater weight to most recent data), with the decay analogous to two time constants in an R-C filter network. Expressed in terms of  $\alpha$ :

$$\text{Decay} = - \frac{2\Delta X_i}{\ln(1-\alpha)}$$

Initially all mines responses were differentiated on TAD plots because the target data was not excluded from the background estimate, in essence reducing the target responses. To correct this, a threshold condition was initially inserted in the determination of the weighting factor  $\alpha$ . In test areas with large background fluctuations, however, the background peaks would also be excluded from the background estimate when the threshold was exceeded.

Average (ASUM) although originally developed as a modifier for the weighting factor to dynamically exclude target responses from the background estimate, the process designated ASUM proved to be a most successful algorithm. ASUM is defined as follows:

$$\text{ASUM}_i = \frac{1}{n} \sum_{j=1}^n \text{An}_{ij}$$

where:

$n$  = number of frequency channels

$\text{An}_{ij}$  = the normalized data at position  $i$  and frequency  $j$

$$\text{An}_{ij} = \frac{A_{ij} - B_{ij}}{\sigma_{ij}}$$

and

$A_{ij}$  = linear value of amplitude data for position  $i$  and frequency  $j$

$B_{ij}$  = exponentially smoothed background estimate for position  $i$  and frequency  $j$ .

$\sigma_{ij}$  = standard deviation, obtained from the exponentially smoothed estimate of the variance.

$B_{ij}$  and  $\sigma_{ij}$  are determined by:

$$B_{ij} = B_{i-1,j} (1-\alpha) + \alpha A_{i-1,j}$$

$$\sigma_{ij}^2 = \sigma_{i-1,j}^2 (1-\alpha) + \alpha (A_{i-1,j} - B_{i-1,j})^2$$

where the weighting factor  $\alpha$  was initially:

$$\alpha_1 = 1 - e^{-2\Delta X_i / \text{decay}}$$

where

$\Delta X_i$  = position increment between samples at  $i$  and  $i-1$

Decay = the distance prior to position  $i$  where the relative weight of the contribution to the background estimate is down to 13%.

The weighting factor was later modified to:

$$\alpha_2 = \begin{cases} \alpha_1; & \text{if } ASUM_i \leq 0.9 \\ 0; & \text{otherwise} \end{cases}$$

and finally to:

$$\alpha_3 = \alpha_1 / (1 + ASUM_i^4)$$

The weighting factor was modified in order to compensate for the effects of varying background levels as a function of soil conditions and moisture conditions.

Many more attempts were made to further improve target discrimination by evaluating statistical methods and methods using magnitude and/or duration of target responses. As a result, six algorithms were developed for evaluation in real-time, real-world field tests with the feasibility model.



4. FEASIBILITY MODEL. To provide real-time test results and flexibility in the selection of test parameters (position and frequency sample intervals, frequency range, etc.), a mine detection feasibility model incorporating advanced discrimination techniques was designed and fabricated. (See block diagram, Figure 4-1). A programmable minicomputer (CDC Model 649) was incorporated into the system to increase the data processing capabilities, an X-Y recorder was added to provide real-time plots of test results, and a four-channel analog instrumentation recorder was employed to record the test data for off-site processing and analysis.

This feasibility system was installed on an electrically powered tractor (G.E. Model E-15) as shown in Figure 4-2. Field tests were conducted (and continue to be conducted) at various field sites, with the purpose of evaluating the real-time discrimination techniques that were previously developed utilizing an off-line system.

The electronic components of the feasibility model consist primarily of a transmitter, a receiver, a System Control Unit (SCU) and the data processing subsystem (CDC 469). As described in the subparagraphs that follow, the receiver compares the phase and amplitude of received signals to the transmitted signal. The resulting phase and amplitude data are examined by the data processing subsystem for indications of the presence of a mine under the detector heads.

4.1 FEASIBILITY SYSTEM OPERATION. The major system components -- transmitter, receiver, system control unit, processor and power supplies -- are housed in an equipment enclosure which mounts to the front of the tractor. The antenna switches mount near the search head at the end of the frame, and the shaft encoder that correlates mine detector data to search head position and operating speed is driven by a rear wheel on the tractor.

The System Control Unit (SCU) responds to synchronizing pulses from the position encoder and control commands generated via the keyboard unit, producing sweep control signals for use by the transmitter. The transmitter signal originates in a voltage-controlled oscillator (VCO) whose output frequency is proportional to input sweep voltage. The VCO output is power-amplified to approximately 1 watt and supplied via a directional coupler and solid-state switch to the transmit antennas, with a -20 dB output of the coupler supplied as a reference to the receiver.

The receive antennas connect via another solid-state switch to the receiver. The SCU synchronously controls the two antennas switches, providing the scanning sequence that selects each transmit-receive antenna pair in turn. The receiver contains a synchronous detector that yields an amplitude output and a phase detector that yields the

phase data signal. Receiver outputs are digitized and processed by the SCU and CPU.

4.1.1 SYSTEM CONTROL UNIT. In addition to generating scan sequence commands in response to position encoder pulses, and generating sweep control voltages and antenna switching commands, the SCU processes amplitude data supplied by the receiver, detects target responses in the processed data, and activates the target alarm and identifies the position of a detected target. In particular, the SCU produces control signals for system operation.

Various computer operational commands and any one of the processing algorithms stored in the computer memory can be addressed via the SCU by means of appropriate keyboard entries. The keyboard is also employed to set up the SCU control circuits which establish the test parameters for a particular operation, as follows:

(1) Frequency Increment - Amplitude data supplied by the receiver is sampled at intervals as the frequency range is swept, and the frequency change during a sample is defined as the frequency increment. The frequency increment is selectable over a range from 10 MHz to 150 MHz in multiples of 10 MHz.

(2) Start Frequency - The start frequency is defined as the frequency chosen for the low end of the swept frequency range, and is selectable in multiples of 10 MHz from 300 to 990 MHz.

(3) Sample Interval - The sample interval is a measure of the duration between each frequency sample, and is selectable over a range from 0.1 ms to 1.5 ms in multiples of 0.1 ms.

(4) Number of Sample Intervals - The number of sample intervals per sweep (up or down) is selectable over a range from 1 to 15 intervals per sweep.

(5) Transition Time - When a sweep reaches the end frequency (either high or low), it dwells at this frequency for a selected time period to allow the antenna switches to select the next antenna pair. This transition time is selectable from 0.1 to 1.5 ms in multiples of 0.1 ms.

(6) Number of Sweeps - The number of sweeps per scan normally corresponds to the number of antenna pairs in the search head array, thus allowing one sweep per pair. The number of sweeps per scan is selectable over a range from 1 to 16.

(7) Scan Rate - The scan rate is the number of complete scans (sweeping all antenna pairs) per unit of travel. The selectable rates are 25, 50, 100 and 200 scans/meter.

4.1.2 CENTRAL PROCESSOR UNIT (CPU). The processor employed in the feasibility model is a CDC Model 469 minicomputer with 8K words of plated wire memory. Word length is 16 bits, and the cycle time is 1  $\mu$ s. There are 42 instructions, and the CPU responds to direct, indexed direct and indirect addressing modes.

The function of the processor is to analyze the digitized data supplied by the SCU and determine, with a high degree of certainty, whether a received response is caused by a mine or some other anomaly in the ground. If the presence of a mine is indicated, an output signal is generated for use in sounding an alarm.

The 8K memory in the processor stores and protects all the processing algorithms, and also contains the operational programs for the SCU. Thus, by means of the control keyboard, any of the various algorithms can be selected, and parameter changes to both processing algorithms and SCU programs can be entered at any time, as desired.

Through use of the computer peripherals, the contents of the memory can be changed as desired by connecting the system to the programmer's console. Changes can then be entered using magnetic tape, paper tape, or via the teleprinter. A monitor oscilloscope is available for displaying the contents of the memory.

5. CONCLUSIONS. Utilizing the feasibility model, a test program was performed in order to evaluate the developed algorithms as well as several detector head configurations. Six algorithms were stored in memory in the CDC 469 minicomputer and could be selected with the keyboard, for operation in the system. Nominal default values were included for the variable test parameters such as decay and position increment as well as for threshold values or conditions applicable to certain algorithms. All these could be changed by addressing a defined location in the CPU by means of the keyboard. In addition, two versions of ASUM were included to establish accuracy requirements: one in floating point and one in 6-place fixed point. The latter provided insufficient accuracy and was later dropped.

From the start it was apparent that performance of any algorithm using one or more thresholds hinged on the margin over which these threshold values could be used at any location and with any soil composition or moisture content. Obviously, under operational conditions, it would not be feasible to obtain a specific threshold value for the encountered soil conditions. One could hardly be expected to operate over a road to establish the required threshold value, then go back over it to detect any possible mines.

After a lengthy, time consuming series of tests at several test sites, it was found that threshold values varied widely with different soil conditions. With the need for a priori knowledge of soil conditions established, the

further use of threshold dependent algorithms were abandoned for the practical reasons stated above.

Real Time field evaluation of the TAD and ASUM algorithms indicated that best performance could be achieved with ASUM (floating-point version).

# FEASIBILITY SYSTEM BLOCK DIAGRAM

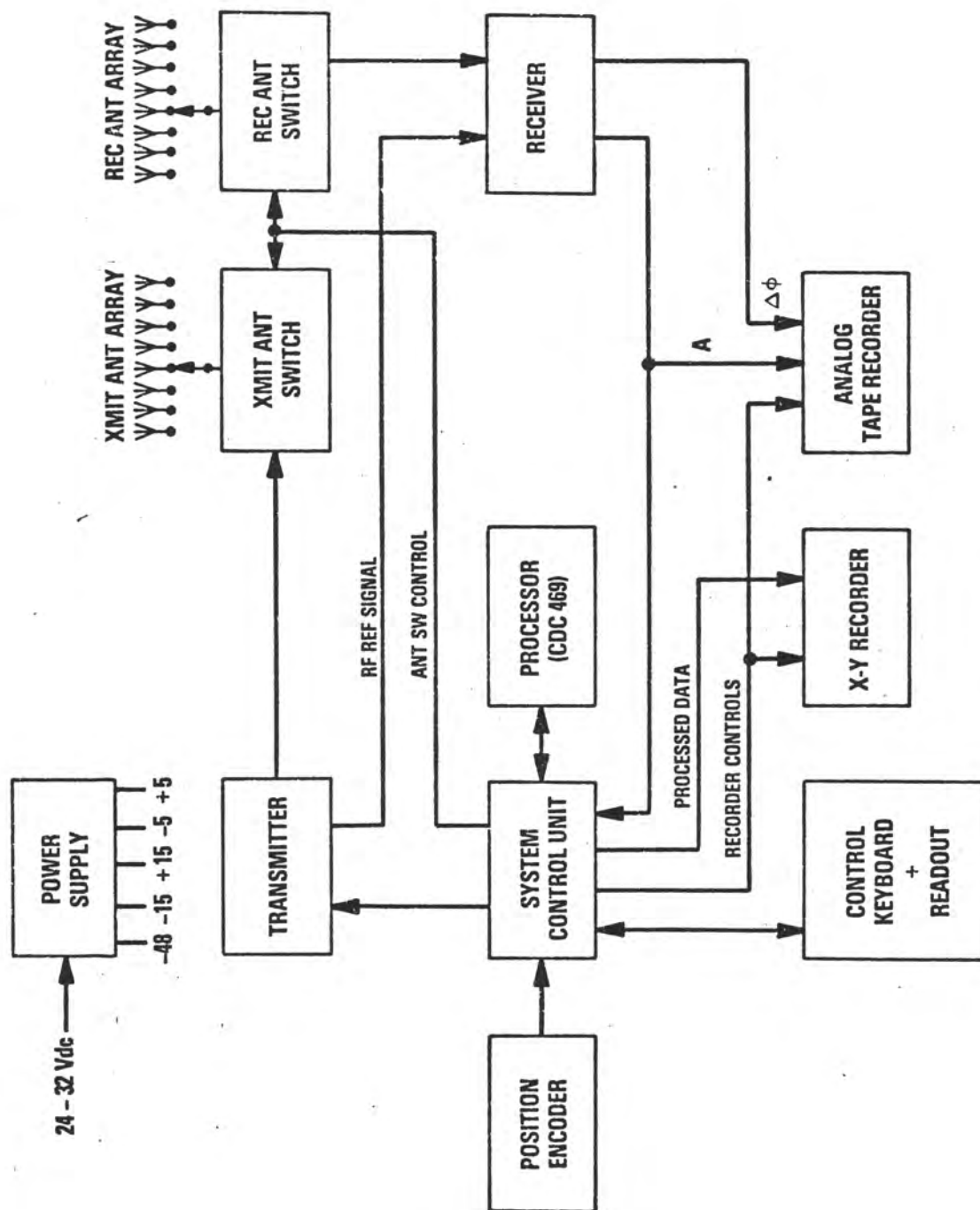


Figure 4-1

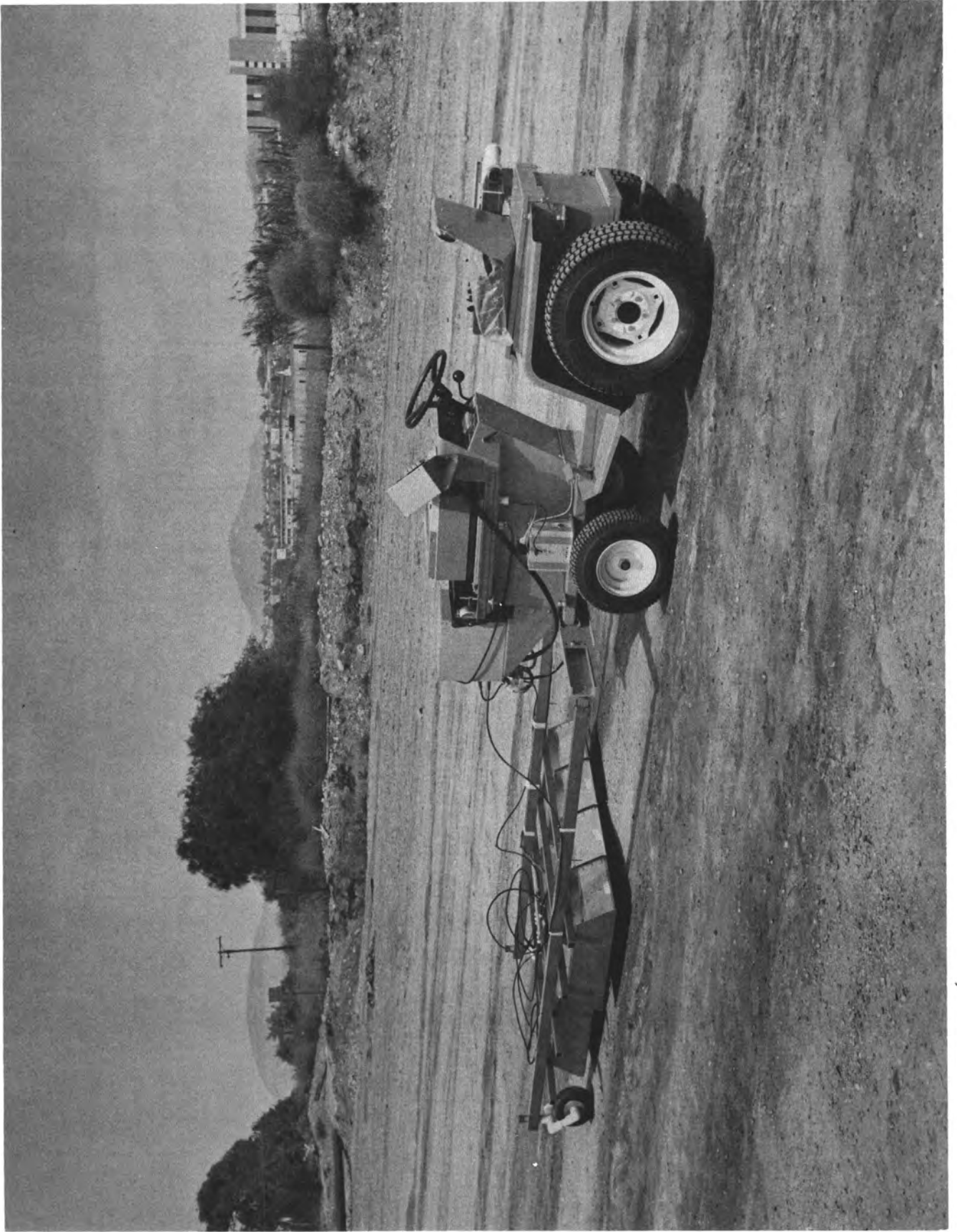


Figure 4-2: Feasibility Model1. Mine Detection System

AUTOMATED CONTROL, DATA ACQUISITION, AND ANALYSES  
FOR HYDRAULIC MODELS OF TIDAL INLETS

D. L. Durham; H. C. Greer, III; and R. W. Whalin  
U. S. Army Engineer Waterways Experiment Station  
P. O. Box 631, Vicksburg, Miss. 39180

ABSTRACT. An Automated Data Acquisition and Control System (ADACS), which was developed (Durham and Greer, 1975) at the Waterways Experiment Station (WES) during the past two years, has been expanded to provide automated control, data acquisition and analyses for hydraulic models of tidal inlets. ADACS configuration consists of a minicomputer with 32K 16-bit words of memory, an interval timer (1  $\mu$ sec), an analog to digital 12-bit converter with 64 analog inputs ( $\pm$  10 volts) and 45 kHz multiplexer, 96 sense/control lines, a magnetic tape controller with two 9-track tape drives, a moving head disk controller with one dual disc drive (removable and non-removable platters), one matrix electrostatic printer/plotter, and an ASR 33 teletype unit.

ADACS controls the hydraulic generation of the tide in the tidal inlet model by providing a programmable analog voltage to the hydraulic tide generator. The programmable tide controller can simulate a tide composed of one to N tidal constituents for as many tidal cycles as required. Model tidal elevations are recorded by ADACS using a bubbler system which measures small hydrostatic pressure changes associated with changes in tidal elevations in the model. The bubbler system consists of a high precision pressure transducer, a scanivalve device for sequencing input ports, and 48 pressure inputs. The pressure transducer can be calibrated prior to and at selected time intervals during each tidal test to provide accurate, updated calibration data for scaling voltage (pressure) data to tidal elevations. In addition to collecting tidal elevation data, tide velocities at specific model locations are monitored by using miniature, electromagnetic current meters. Besides controlling the tide generator during a tidal test, ADACS acquires the above tidal data (elevations and velocities) including calibration information and

test parameters and records these data on magnetic tape for future analyses and permanent storage.

Data analyses include scaling and editing of original data, least squares harmonic analyses of tidal data (elevation and velocity) for amplitude and phase of various tidal constituents, plots of original data and superimposed harmonic constituents, and analyses of residual variances. Verification of hydraulic tidal inlet models, which are geometrically distorted models scaled on the Froude model law, requires artificially simulating in the model the frictional effects associated with prototype roughness of bottom and sides of channels and overbank and marsh land roughness. The relative phase lags of major tidal constituents from one specific location to another in the prototype and the measurement of these phase lags in the model provide a means for estimating the amount of roughness and specific model areas requiring artificial roughness in order to achieve model verification.

These procedures for spatial definition of roughness, full model verification, and model testing using ADACS and associated automation procedures have been successfully applied in hydraulically modeling Murrells Inlet, South Carolina. The required time for model verification and testing, as well as data analyses, has been significantly reduced. In addition, the quality and quantity of model data has increased with minimal cost increases. Thus, better information can be made available on which to make sound engineering decisions regarding the planning and design of proposed engineering changes in tidal inlets.

1. Introduction. Over the past decade, automated processing technology has evolved from large, expensive computers to minicomputers and microprocessors. With this evolution, the physical size and cost of automated processing systems have greatly decreased with a minimal decrease in system capabilities. Cost reductions for such systems have resulted in economic justification of the use of minicomputer and microprocessors to a specific task or group of specific operations; whereas, large computers can be justified economically only for multiple operations



and tasks. The automation of physical, hydraulic modeling techniques has lagged automation efforts in many other fields mainly because of cost justification and the requirements of highly specialized instrumentation (e.g. sensors). However, needs for such automation have existed for many years. These needs are the result of requirements for (1) real-time model control decisions, (2) quasi real-time data analyses, and (3) more accurate and reliable model data for engineering and environmental interest studies.

One mission of the Hydraulics Laboratory of the U. S. Army Engineer Waterways Experiment Station is the physical modeling of hydraulic problems associated with the activities of the Corps of Engineers, as well as other government and private agencies. Hydraulic problems associated with wave phenomena and the effects of these phenomena in harbors, tidal inlets, and along the open coast are the primary modeling interests of the Wave Dynamics Division (WDD) of Hydraulics Laboratory. Over the last three years, WDD has been very successful in automating<sup>1,2</sup> the major aspects of its physical models for wave and tidal inlet studies. Major automation efforts were devoted to (1) model control, (2) model data acquisition, and (3) model data reduction and analyses. The subject of this paper is a description of the automated system, which has been given the name "Automated Data Acquisition and Control System" (whose acronym is ADACS), for wave and tidal models and its general application to tidal inlet model studies.

2. SYSTEM CONFIGURATION. The automated system for wave and tidal inlet models has two primary functions: (1) automated acquisition of wave and tide data in a format (magnetic tape or disc) compatible for digital reduction and analyses and (2) automated control of model sensor calibration and of the wave and tide generators. The design, development, and configuration of automated systems to perform these functions with particular applications to wave models were presented in Reference 1. System configuration (Fig. 1) of ADACS consists basically of the following four subsystems:



- a. Digital data recording and controls.
- b. Analog recorders and channel selection circuits.
- c. Wave/tide sensors and interfacing equipment.
- d. Wave/tide generators and control equipment.

The first subsystem is basically a 32 K, 16-bit word minicomputer with I/O and storage devices, analog/digital packages, and a timing package. Details of this subsystem were presented by Durham and Greer (1975). The analog recording subsystem is (1) a backup for the digital data recording subsystem and (2) a visual display for operator inspection of analog signals from model sensors. This subsystem has manual/automated selection and control of five, 12-channel oscillographs and a test point center for manually monitoring a selected channel as to system setup, calibration, and signal condition.

The model sensor subsystem includes instrumentation for both wave and tide sensors. Details of the wave sensor subsystem and calibration procedures were presented in Reference 1. The wave sensor subsystem consists basically of the following four components:

- a. Wave height sensors and stands.
- b. Power supplies and signal conditioning equipment.
- c. Manual and automatic calibration equipment.

The sensor subsystem for tidal heights includes the following major components:

- a. Bubble tubes, stands, and high pressure supply.
- b. Scanivalve with manual and automated controls.
- c. Precision pressure transducer.
- d. Power supplies and signal conditioning equipment.

The last subsystem includes controls for both wave and tide generators. Controls for both mechanical and electrohydraulic wave generators can be provided by ADACS. Start/stop commands are available for mechanically gear-driven wave generators; however, controls for wave period and

amplitude must be supplied manually. For the electrohydraulic wave generator, ADACS provides a programmable analog voltage as a command signal to the servocontroller of the electrohydraulic actuators. The wave period and amplitude for these wave generators are controlled by ADACS.

The command signal to the tide generator is a programmable analog voltage which is supplied by ADACS through one channel of the digital to analog converter. The tide generator has the option of receiving this command signal from ADACS or accepting an analog voltage from a programmable cam and reference potentiometer arrangement. This latter control scheme is used as a back-up or alternate control to the ADACS control and until recently has been the primary control of the tide generator prior to installation of ADACS. In addition to the command signal, the tide generator has four other major components which are (a) differential amplifier and power supply, (b) bubble tube positioner, (c) hydraulic-pneumatic amplifier, and (d) hydraulic cyclinder and flow-control gate assembly.

Basically, the tide in a physical model is generated from cyclic exchanging by controlled flow a predetermined volume of water between the physical model and a tidal reservoir (sump). Figure 2 is a schematic of the tide generator and controls. The programmed command signal causes a change in the vertical position of the bubble tube relative to the water level in the model. This position change perturbs the equilibrium position of the pneumatic-hydraulic amplifier and results in a differential hydraulic pressure applied to the hydraulic cyclinder activating the flow-control gate. The movement of the flow-control gate is in a direction to correct the perturbed equilibrium condition of the pneumatic-hydraulic amplifier by changing the water-surface elevation in the tide model. A feedback circuit from the hydraulic cyclinder to the differential amplifier/bubble tube positioner provides a "damping effect" to prevent gate overshoot and unstable oscillations. Thus, any tidal constituent or progressive tide can be used as the forcing function for the tide

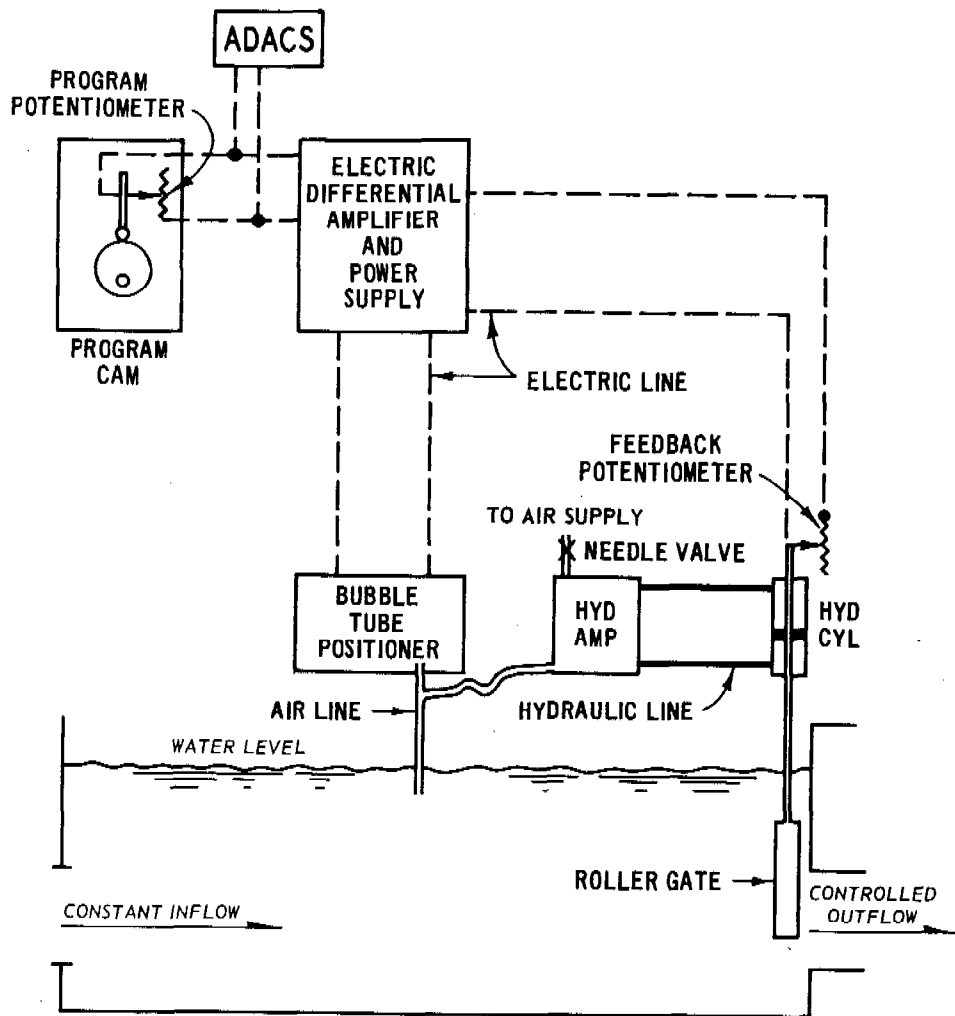


Figure 2. TIDE GENERATOR AND CONTROLS

model by programming ADACS to produce a command signal harmonically representing the appropriate forcing function.

3. TIDAL HEIGHT SENSORS. Data, which are acquired by ADACS from tidal inlet models, consist of time histories of water surface variations relative to some reference water level. For specified tide conditions at the generator, tidal elevations are collected at selected locations within the tidal model. These data are used to calculate mean tide levels, tidal ranges, arrival times of high and low water, and the phases and amplitudes of specific tidal constituents. Although various types of tidal height sensors are used by the Hydraulics Laboratory, a tide sensor system developed and implemented within the last year for use in tidal inlet models is presented in this paper. For lack of a better name, this sensor subsystem has been labeled the "bubbler system." This system measures small hydrostatic pressure changes associated with changes in tidal elevations in the model and consists of a high precision, pressure transducer, a scanivalve device for sequencing input ports, and 48 pressure inputs.

To employ the bubbler system (Fig. 3), a small plastic tube is inserted some small distance into the water. The outside diameter of this bubble tube is required to be small to minimize blockage of tidal flow, etc. The tube is connected through a throttling valve to a regulated pressure supply. This valve or restriction serves to regulate air flow and isolate the bubble tube from other bubble tubes and the supply pressure. Each bubble tube is connected to a common pressure transducer, which is parallel with the throttling valve and high pressure source, of suitable pressure range to accurately detect the tube's internal pressure changes associated with changes of water surface elevation.

To allow many bubble tubes to share a common pressure transducer, a multiplexing device, which is called a scanivalve, is used. This device multiplexes sequentially many pressure inputs to a common output port. A high precision pressure transducer is included as an integral part of the output port. A controller is used to advance the unidirectional

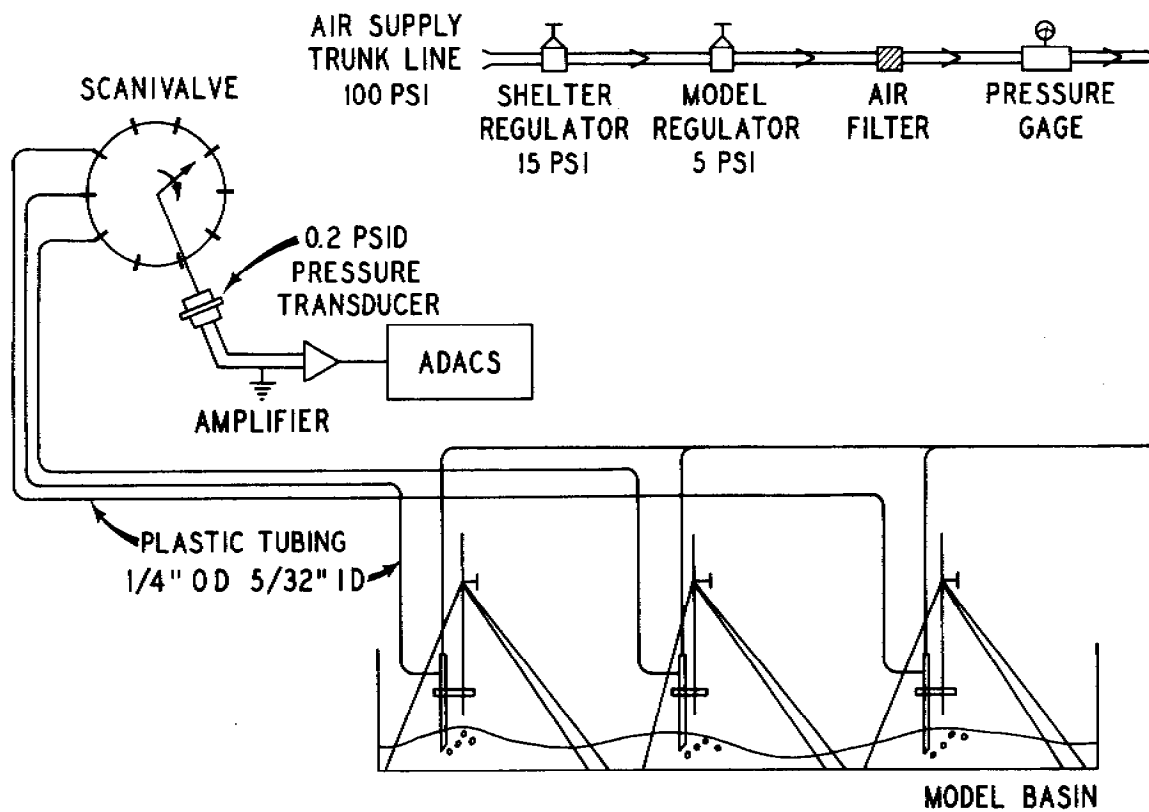


Figure 3. BUBBLER SYSTEM

stepping motor which increments the scanivalve. By contact closures or commands from ADACS, the valve can be advanced sequentially to any input port or to a "home" (reference) position without intermediate stops.

The system presently used by the Wave Dynamics Division is capable of accepting up to 48 pressure inputs and includes a  $\pm 0.25$  psid pressure transducer with a nonlinearity and hysteresis (best straight line) of 0.05 percent full scale. The pressure transducer output is an analog voltage of  $\pm 10$  volts full scale. The pressure cell is interfaced through appropriate signal conditioning equipment to the analog multiplexer of the digital recording subsystem. The valve accepts both home and step commands from the ADACS and has a BCD position feedback to the ADACS. The system can be completely controlled by either ADACS or manual controls.

To install the bubbler system in the model, the water level in the tidal model is raised to mean higher high water (MHHW). At this still water level, the orifice of the bubbler is inserted into the water to a depth which is slightly greater than the maximum expected tidal range (hydrostatic head). At this elevation, the pressure supply must be set high enough to cause the emission of air bubbles from the tube. For these conditions, the tube's internal pressure is equal to the hydrostatic pressure of the water column above the orifice of the bubble tube. The tube orifice is cut diagonally to aid in the bubble's escape. It is important that the system bubble freely at this depth because the tube's internal pressure ceases to be equal to the hydrostatic head with bubble cessation. At this point, the bubble tubes should be observed over several tidal cycles to be certain there is a continuous stream of bubbles.

The bubbler system with the arrangement of bubble tubes, scanivalve, and a high precision pressure transducer provides a very economical system of obtaining precise measurements of water surface elevations at a large number of locations in a tidal model. Tidal elevation measurements by this system are accurate to 0.001 feet. The sampling sequence of the scanivalve is rated at a maximum of 10 samples per second. This method of detecting changes of water surface elevation is limited only by the



three constant pressure values over the tidal range. These values are obtained by setting a bubble tube at each of the following three tide levels: mean lower low water, mean tide level, and mean higher high water. The three bubble tubes are positioned to these levels in a stilling basin which is connected to the tidal model by a cut-off valve. Prior to each tidal test, the water level in the model and stilling basin are raised to mean higher high water. The stilling basin is then isolated from the model by closing the cut-off valve. Finally, the three bubble tubes are adjusted to their appropriate water depth. Throughout the tidal test these bubble tubes are monitored at every scan to provide update calibration data. During data analysis, calibration information can be updated by calculating calibration coefficients for each scan or any multiple of scans.

A limited number of channels of tidal velocity can be measured by miniature, electromagnetic current meters which are monitored by ADACS. The collection of tidal velocities using ADACS has not been fully implemented at this time and is pending the completion of transducer evaluation which should be completed within the next year. Until such time, the majority of tidal velocity measurements are obtained manually by using a modified version of the miniature Price meters.

In addition to tide data, many tidal inlet studies require wave information as well. The generation of waves and collection of wave data at specific tidal phases (normally high, low, and mean tide levels) are provided by ADACS. While controlling the tide generator and collecting tidal data, ADACS uses in-core timers to determine the occurrence of specified tidal phases at which times (1) the wave generators are turned on, (2) wave data at a specified sampling rate for a predetermined number of wave periods are collected at various locations in the model, (3) the completion of wave test for that tidal phase is detected, (4) the wave generators are turned off, and (5) in-core timers initialized to determine the next specified tidal phase for wave tests. These wave tests are performed normally during the middle cycle of a three-cycle tidal test. The instrumentation and procedure for collecting wave data are the same as described in Reference 1.

frequency response of the system and the accuracy of the pressure transducer. Application of this system to tidal models has resulted in large dollar savings when 5 or more locations in a model are instrumented.

4. DATA ACQUISITION. During the acquisition mode, tidal data for a programmed tidal condition at the generator are collected from a specified number of tide sensors, digitized, and recorded on magnetic tape or disc for further analyses. The sampling scheme is flexible and can be tailored for different applications with maximum thru-put rates theoretically limited by the multiplexing rate of the scanivalve. The present sampling scheme is to (a) increment the scanivalve to the first data channel, (b) delay a specified time interval (normally 0.5 sec) to allow input pressure to stabilize, (c) collect a specified number (normally 10) of samples, (d) average these voltage samples, (e) store the discrete sample in memory, (f) increment to the next channel, (g) repeat the above procedure, and (h) continue sequentially through remaining channels. For each tide sensor, 100 discrete voltage samples are collected at equally spaced intervals over each tidal cycle for a predetermined number of cycles (normally 3 to 5). The minicomputer calculates from input parameters (1) the required timing interval between multiplexing scans of the scanivalve to provide the correct sampling rate, (2) the delay interval at each channel, and (3) the number of voltage samples to be digitized and averaged and initializes counters for determining completion of tidal tests. In addition, it provides an analog command signal through the digital to analog converter to the tide generator and lags the beginning of data acquisition by a specified number of tide cycles after starting the generator.

Due to thermal effects (zero drift) on the transducer output over a tidal test of 2 to 3 hours duration, the pressure transducer is calibrated prior to and at selected time intervals during each tidal test to provide accurate, update calibration data for scaling voltage (pressure) to tidal elevations. The calibration data are obtained by monitoring

At completion of the acquisition mode, the calibration, wave, and tide data have been recorded in binary form on magnetic tape or disc. These data with a header for test identification and pertinent parameters are available from disc or magnetic tape for analyses.

5. DATA ANALYSES. Analyses of the elevation and velocity data from tidal model are performed by either the minicomputer subsystem or a Honeywell G635 of the Automated Data Processing Center at WES. Schematically, the automated procedures for analyzing tidal data are as follow:

I. Program Initialization

- (1) Input test parameters and option flags.
- (2) Read and decode data tape or disc file.
- (3) Demultiplex data files and scale data.

II. Tidal Data Analyses

- (1) Harmonic analysis using Least Squares techniques.
  - (a) Amplitude and phases of tidal constituents.
  - (b) Relative phases between gages.
- (2) Analyses of residual variances.
  - (a) Original versus Least Square estimate.
  - (b) Prototype tide versus model tide.
  - (c) Model base test versus model plans.
- (3) Graphic output of above results.

In addition to the above automated procedures, manual and photographic techniques are employed in tidal models to study general patterns of tidal circulation and to define qualitatively littoral transport and deposition patterns.

The analyses of data from wave models are presented by Durham and Greer (1975) and are basically auto-spectral and cross-spectral analyses, statistical analyses for wave heights and periods of wave signals at selected locations throughout the model, and computation of response functions or amplification factors from wave energy within the harbor or tidal inlet relative to incoming wave energy. In addition to

analyzing data from wave models, these procedures are used also in analyzing wave data which are generated and acquired (refer to previous section) at selected tidal phases and/or tidal ranges during specific tide/wave tests in the tidal model.

6. MODEL APPLICATION. The hydraulic tidal model is used as an engineering tool in predicting the effects of proposed engineering changes (channel dredging, inlet geometry changes, and construction of coastal structures) and in planning and designing the proposed changes in a cost effective and environmentally compatible manner. For physical models of tidal inlets to be most effective as a predictive tool, verification of such models is desirable. Verification of hydraulic tidal inlet models requires the hydraulic model to reproduce observed prototype conditions of tidal elevations, tidal phases, and average mass distributions at specified locations and/or cross sections in the model. Hydraulic tidal inlet models are scaled on the Froude model law and are usually geometrically distorted models in which the horizontal length scale is not the same as the vertical length scale. Such modeling procedures require artificially simulating in the model the frictional effects associated with prototype roughness of bottom and sides of channels and overbank and marshland roughness.

Various procedures are available for inducing artificially the frictional effects<sup>3,4</sup> in the hydraulic tidal model. One such procedure is the use of roughness (drag) elements which are small strips of metal that are attached vertically to the model bottom. The practice of using such roughness strips for simulating frictional effects in tidal inlet models is a standard and accepted procedure in physical modeling. The successful application (required number and horizontal distribution of roughness strips) of this procedure most frequently depends upon the modeling experience of the hydraulic engineer and requires lengthy testing programs. A means of estimating theoretically and/or empirically the required number of roughness elements and their horizontal distribution

is a needed capability in physical modeling. One method of estimating the uniform horizontal distribution (number of elements per square foot) of roughness elements in a distorted hydraulic model has been proposed by Multer.<sup>5</sup> Although this procedure is based on one-dimensional tidal flow, it warrants additional laboratory investigation and study. One desirable refinement is to be able to define the variation in the horizontal distribution of roughness elements from an idealized uniform horizontal distribution. The relative phase of the tide at different locations along a tidal channel relative to a reference location in the channel can be used as one parameter to estimate the horizontal distribution of roughness elements. Having placed a uniform horizontal distribution of roughness elements in a tidal hydraulic model, comparison of the model and prototype relative phases of a dominant tidal constituent between two locations along a specific channel can provide information as to the change in the amount of model roughness required to reproduce correctly the prototype tidal phase along the specific reach of channel between the two chosen points. Thus, the relative phase lags or leads of major tidal constituents between selected locations throughout the tidal inlet can provide similar information for the various channel reaches in the model. If sufficient prototype data (amplitudes and phases of various tidal constituents) are available, much information and guidance as to the horizontal distribution of roughness elements in the tidal hydraulic model can be obtained from the above procedure.

These procedures for defining the horizontal distribution of model roughness, full model verification, and model testing using ADACS and associated automation procedures were successfully applied in hydraulically modeling Murrells Inlet, South Carolina. A fixed-bed tidal model study was conducted to estimate the effects of proposed engineering changes in the mouth of the inlet on the tide and wave regime in the inlet. The prototype area of interest was approximately 1.5 miles inland and 6.5 miles along shore. A distorted physical model with an area of approximately 21,800 sq ft was scaled at 1 to 60 vertically and 1 to 100 horizontally and constructed with molded topography reproduced

to -25 ft offshore and +10 ft on land. Good prototype data was available from National Ocean Survey of NOAA for 8 tide gages within the tidal inlet. Figure 4 is a schematic of the model and the locations of the various tide gages. The tidal regime of this inlet is dominated by the principal lunar semidiurnal constituent,  $M_2$ , whose variance represents over 85 percent of the tidal variance in the inlet.

In the verification of Murrells Inlet model, the major tidal constituent,  $M_2$ , and its overtides,  $M_4$ ,  $M_6$ , and  $M_8$  were used in the initial verification tests. After estimating and placing a quasi-uniform distribution of roughness strips in the model, the tide generated in the model was composed of the  $M_2$ ,  $M_4$ ,  $M_6$ , and  $M_8$  tidal constituents. The tidal coefficients (amplitudes and phases) for these and 20 other tidal constituents were calculated by National Ocean Survey from prototype data collected at the 7 tidal gages within Murrells Inlet and one tide gage just outside the mouth of the inlet. The instantaneous height (elevation) of the prototype tide at a specific location in the inlet can be represented by the following equation.

TIDAL HEIGHT:

$$h(t) = H_0 + \sum_{i=1}^N f_i H_i \cos [a_i t + (V_0 + u)_i - \kappa_i]$$

WHERE

$h$  = Tidal height at time  $t_i$ .

$H_0$  = Mean height above reference datum.

$H_i$  = Mean amplitude of  $i^{\text{th}}$  constituent.

$f_i$  = Factor to reduce mean amplitude to year of prediction

$a_i$  = Angular speed of  $i^{\text{th}}$  constituent.

$t$  = Time reckoned from some initial epoch.

$(V_0 + u)_i$  = Equilibrium argument of  $i^{\text{th}}$  constituent for  $t=0$ .

$\kappa_i$  = Local epoch of  $i^{\text{th}}$  constituent.

$N$  = Total number of constituents.

Harmonic analysis performed by National Ocean Survey on prototype data provide  $H_0$ ,  $H_i$ , and  $\kappa_i$  for each tidal gage. The other coefficients,

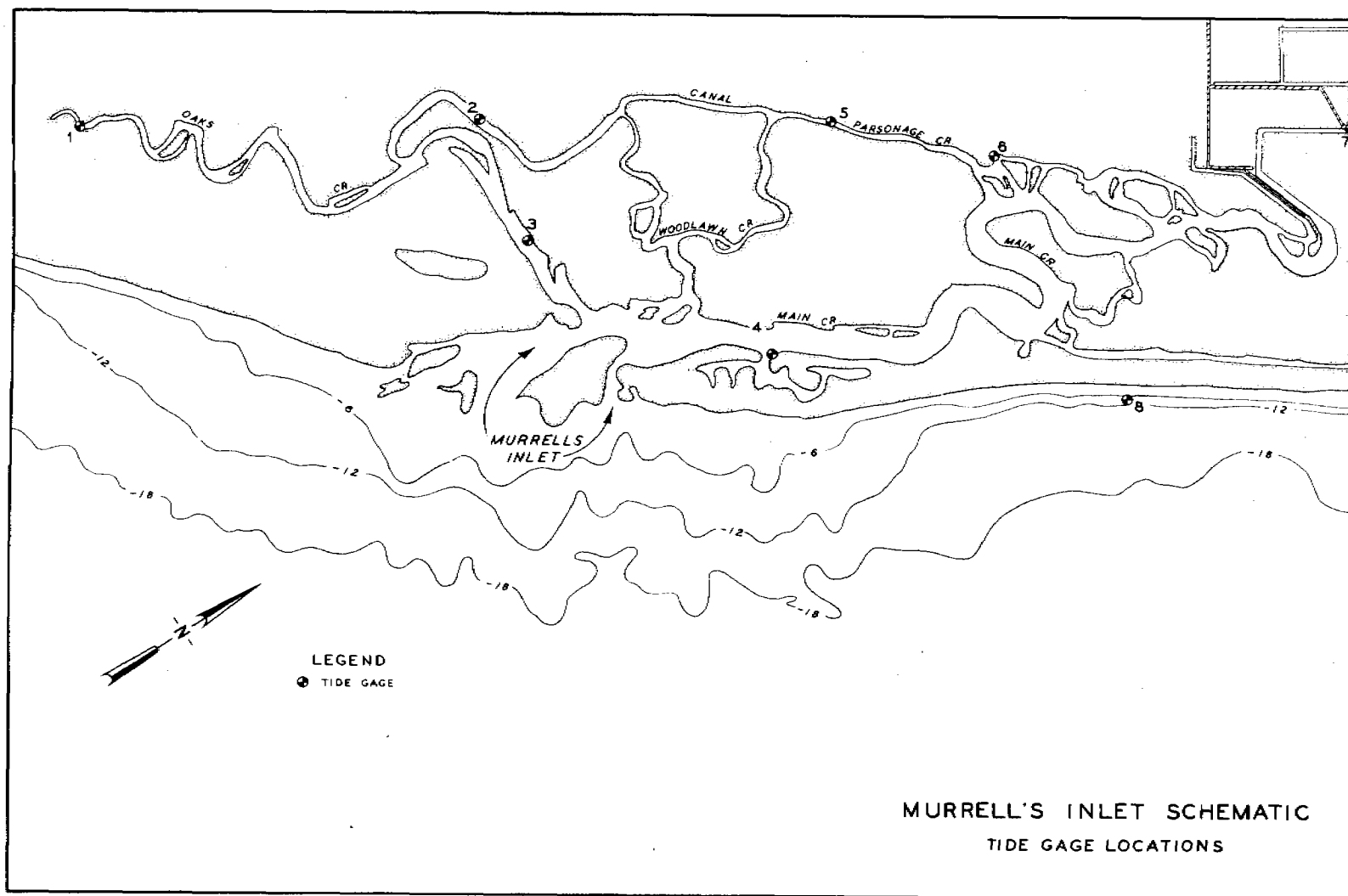


Figure 4. Schematic of Model and Gage Locations

$f_i$ ,  $a_i$ , and  $(V_0+u)_i$ , can be obtained from appropriate tables.<sup>7</sup> The preceding equation can be rewritten in the following form:

$$h(t) = H_0 + \sum_{i=1}^N A_i \cos(\omega_i t + \phi_i)$$

WHERE

$h$  = Tidal height at time  $t$ .

$H_0$  = Mean height above reference datum

$A_i = f_i H_i$  = Amplitude of  $i^{\text{th}}$  constituent.

$\phi_i = (V_0+u)_i - \kappa_i$  = Phase of  $i^{\text{th}}$  constituent.

This form of the instantaneous tidal height is used in model control and data analyses of the hydraulic tidal model. For Murrells Inlet, the harmonic function composed of the  $M_2$ ,  $M_4$ ,  $M_6$ , and  $M_8$  tidal constituents at Gage 8 was used as a command signal to the tide generator in producing the model tide forcing function for the initial verification tests.

At each of the gage locations, the instantaneous tidal heights were recorded simultaneously for several tidal cycles. The instantaneous tidal height,  $h_M(t)$ , at each gage location in the model can be represented as follows:

$$h_M(t) = \hat{h}_M(t) + \epsilon(t) \equiv a_0 + \sum_{i=1}^N [a_i \cos(\omega_i t) + b_i \sin(\omega_i t)] + \epsilon(t)$$

where  $\hat{h}_M(t)$  is the calculated tidal height, which is represented by a harmonic series of known frequencies, and  $\epsilon(t)$  is noise associated with the transfer function of the hydraulic model, etc. in the tidal record. With an undetermined noise level in the tidal height record, the principal of least squares can be used to solve for the unknown coefficients (amplitudes and phases) for the  $M_2$ ,  $M_4$ ,  $M_6$ , and  $M_8$  tidal constituents by minimizing the variance or the sum of the squared differences between the measured model tidal height and the assumed form of the model tidal height. The application of the principal of least squares is as follows:



$$E \equiv \int_T \epsilon^2(t) dt = \int_T [h_M(t) - \hat{h}_M(t)]^2 dt \rightarrow \text{MINIMAL}$$

T = RECORD LENGTH

REQUIRES

$$\frac{\partial E}{\partial a_0} = 0; \quad \frac{\partial E}{\partial a_i} = 0; \quad \text{AND} \quad \frac{\partial E}{\partial b_i} = 0$$

Thus, have  $2N+1$  normal equations for  $a_0$ ,  $a_i$ , and  $b_i$  where  $i=1, \dots, N$

$$[C] [A] = [F]$$

C = Real, symmetric matrix of sine and cosine products

A = Vector of coefficients

F = Vector of products of observed signal and sine or cosine terms

N = Total number of constituents

The above set of equations can be solved by routine procedures of matrix inversion. With the model data in digital form from ADACS, this method of analysis is very easily performed by ADACS.

Having obtained the harmonic coefficients for the model tidal height at each tide gage, the relative phases of the tidal constituents at each gage relative to the tidal gage (Gage #8) at the inlet mouth and the differences in these relative phases in the model and the prototype can be determined. Tables 1 and 2 give the amplitudes and model to prototype differences in relative phases of the  $M_2$  tidal constituent for model tests #1, #36, and #73. In addition, the relative phases between any two tidal gages can be obtained, and the difference of these relative phases in prototype and model can be calculated. Table 3 gives the model to prototype differences of the relative phases of the  $M_2$  tidal constituent for appropriate gages for model tests #1, #36, and #73. In model test #1, quasi-uniform horizontal distribution of roughness elements was placed in the hydraulic model of Murrells Inlet. From Tables 2 and 3, the channel reaches (model area between two gages) requiring more or less roughness elements can be determined by considering the magnitude and sign of the model to prototype differences in relative

Table 1

## M2 CONSTITUENT TIDAL AMPLITUDES (FT)

STATION	PROTOTYPE	RUN NO. 1		RUN NO. 36		RUN NO. 73	
		MODEL	DIFFERENCE	MODEL	DIFFERENCE	MODEL	DIFFERENCE
1	1.788	2.150	+0.362	1.804	+0.016	1.716	-0.072
2	1.834	2.140	+0.306	1.922	+0.088	1.789	-0.045
3	1.866	2.070	+0.204	1.916	+0.050	1.797	-0.069
4	1.919	2.260	+0.341	1.974	+0.055	1.878	-0.041
5	1.885	2.220	+0.335	1.957	+0.072	1.838	-0.047
6	1.936	2.270	+0.334	1.986	+0.050	1.872	-0.064
7	1.865	1.950	+0.085	1.885	+0.020	1.819	-0.066
8	2.402	2.430	+0.028	2.412	+0.010	2.396	-0.006

Table 2

## M2 CONSTITUENT PHASE DIFFERENCES (DEG\*)

STATION TO STATION	PROTOTYPE	RUN NO. 1		RUN NO. 36		RUN NO. 73	
		MODEL	DIFFERENCE	MODEL	DIFFERENCE	MODEL	DIFFERENCE
8 - 1	48.23	30.50	-17.73	45.14	-3.09	49.24	+1.01
8 - 2	33.40	26.50	-6.90	32.20	-1.20	34.20	+0.80
8 - 3	19.52	13.80	-5.72	19.20	-0.32	20.20	+0.68
8 - 4	20.69	13.80	-6.89	21.40	+0.71	20.30	-0.39
8 - 5	32.66	23.90	-8.76	32.30	-0.36	33.10	+0.44
8 - 6	33.23	21.20	-12.03	32.00	-1.23	32.50	-0.73
8 - 7	47.09	41.00	-6.09	44.46	-2.63	46.80	-0.29

\* 1 DEGREE = 2.07 MINUTES OF PROTOTYPE TIME

Table 3

## M2 CONSTITUENT PHASE DIFFERENCES (DEG\*)

STATION TO STATION	PROTOTYPE	RUN NO. 1		RUN NO. 36		RUN NO. 73	
		MODEL	DIFFERENCE	MODEL	DIFFERENCE	MODEL	DIFFERENCE
2 - 1	14.83	4.60	-10.23	12.94	-1.89	15.04	+0.21
3 - 1	28.71	17.30	-11.41	25.94	-2.77	29.04	+0.33
3 - 2	13.88	12.70	- 1.18	13.00	-0.88	14.00	+0.12
4 - 5	11.98	10.10	- 1.88	10.90	-1.08	12.80	+0.82
4 - 6	12.54	7.40	- 5.14	10.60	-1.94	12.20	-0.34
4 - 7	26.38	27.20	+ 0.82	23.06	-3.32	26.50	+0.12
5 - 6	0.57	-2.70	- 3.27	-0.30	-0.87	-0.60	-1.17
6 - 7	13.84	19.70	+ 5.86	12.46	-1.38	14.30	+0.46

\* 1 DEGREE = 2.07 MINUTES OF PROTOTYPE TIME

Table 4

M2 CONSTITUENT MEAN TIDE LEVELS  
(FEET ABOVE MLW)

STATION	PROTOTYPE	RUN NO. 1		RUN NO. 36		RUN NO. 73	
		MODEL	DIFFERENCE	MODEL	DIFFERENCE	MODEL	DIFFERENCE
1	2.668	2.833	+0.165	2.499	-0.169	2.734	+0.046
2	2.674	2.831	+0.157	2.427	-0.247	2.695	+0.020
3	2.696	2.632	-0.062	2.249	-0.447	2.546	-0.150
4	2.715	2.733	+0.018	2.380	-0.335	2.663	-0.052
5	2.696	2.794	+0.098	2.423	-0.273	2.682	-0.014
6	2.680	2.755	+0.075	2.387	-0.293	2.655	-0.025
7	2.676	3.012	+0.336	2.523	-0.153	2.762	+0.086
8	2.344	2.557	+0.213	2.128	-0.216	2.365	+0.021

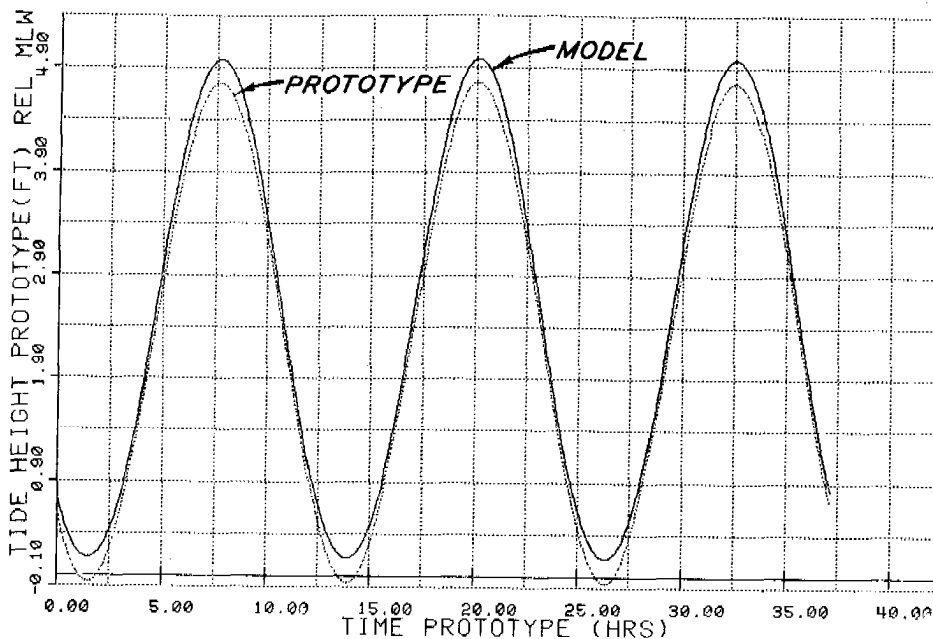
phases between specific gages. In addition to relative phase differences, the tidal amplitudes (Table 1) and mean tide levels (Table 4) must be considered in adjusting the model forcing function and the model roughness during verification. After model test #73, the hydraulic model is considered verified for the  $M_2$  tidal constituent and its overtides. Tidal amplitudes and mean tidal levels in the model agree to within 0.1 feet of prototype measurements and relative phase differences have been reduced to one degree or less. Figures 5 and 6 show plots of model and prototype  $M_2$  tidal heights before and after model verification at Gages 8 and 2, respectively. These plots show that corrections for mean tide levels, tidal ranges, and tidal phases were performed during verification in getting the hydraulic tidal model to reproduce prototype conditions.

The above verification procedure is repeated using a progressive tide as the model tidal forcing function. Since the  $M_2$  tidal constituent represents more than 85 percent of the tidal variance in Murrells Inlet, minimum adjustments to the model roughness are required for model verification using a progressive tide. The harmonic analyses of a progressive tide requires a much longer tide record (approximately 15 days) than the harmonic analyses of the  $M_2$  tidal constituent and its overtides. Thus, verifying initially with the  $M_2$  tide (dominant tidal constituent) reduces the total time required for model verification. In addition, using the relative phase differences to isolate model areas requiring changes in roughness elements has decreased the total time required to verify a hydraulic tidal model.

Velocity verification can be performed in the hydraulic model using the same procedure as presented above for the tidal heights. Frequently, insufficient data exist for the tidal velocity regime to be analyzed for the various tidal constituents. In most cases, twenty-four hours of tidal velocity data exist for appropriate cross sections in the various channels of the tidal inlet. For such cases, a progressive tide can be generated in the model after tidal height verification. This progressive

## BEFORE VERIFICATION

MURRELLS INLET RUN NUMBER 1 GAGE NUMBER M-08



## AFTER VERIFICATION

MURRELLS INLET RUN NUMBER 73 GAGE NUMBER M-08

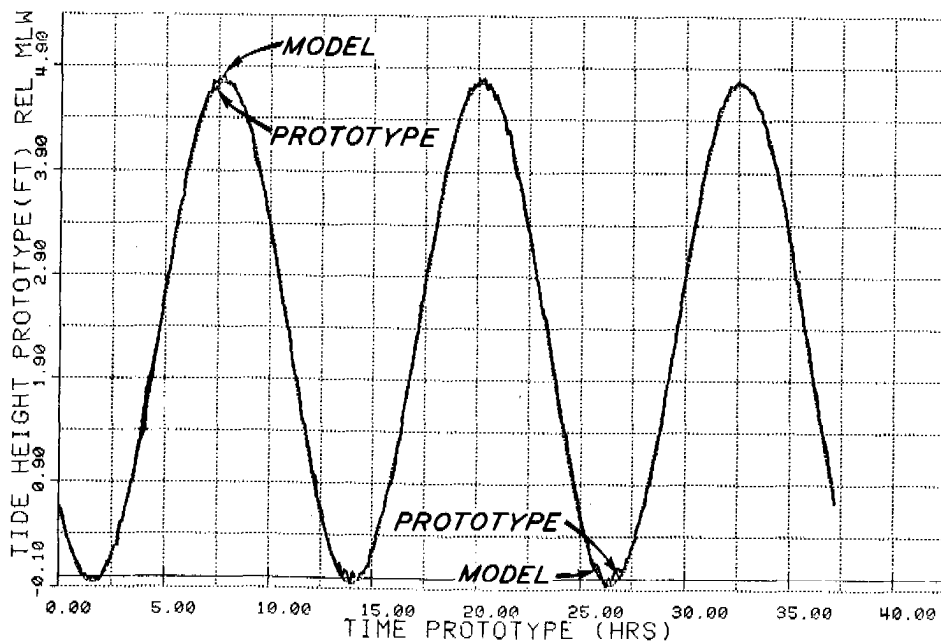
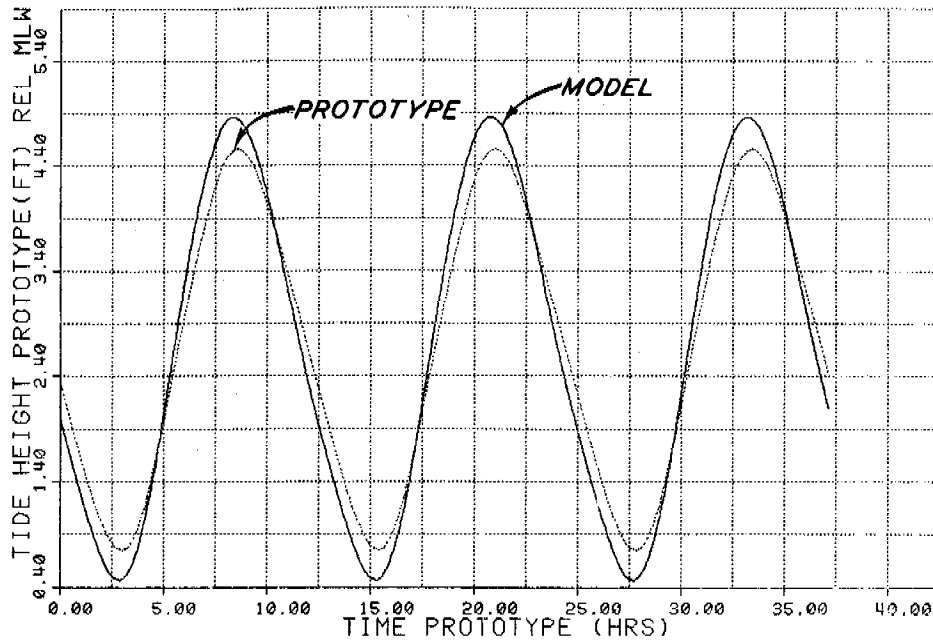


Figure 5.  $M_2$  Tidal Elevation Near Mouth of Murrells Inlet.

## BEFORE VERIFICATION

MURRELLS INLET RUN NUMBER 1 GAGE NUMBER M-02



## AFTER VERIFICATION

MURRELLS INLET RUN NUMBER 73 GAGE NUMBER M-02

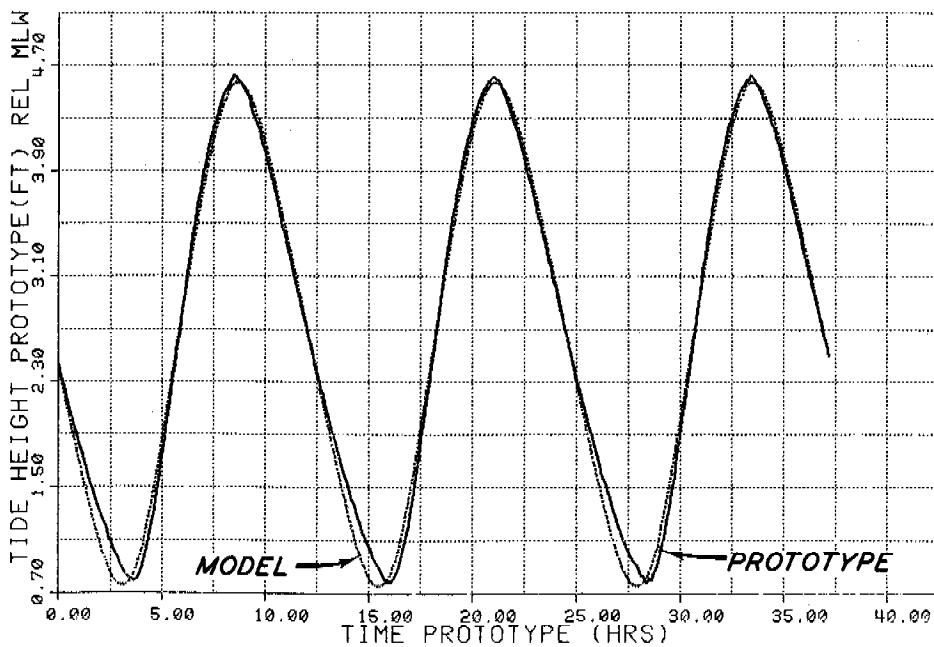


Figure 6.  $M_2$  Tidal Elevation in Back Reach of Estuary.

tide can be made to correspond to the appropriate progressive tide for the particular twenty-four hours of interest. Then, tidal velocity measurements and required model adjustments can be performed to obtain model reproduction of prototype tidal velocity.

The first application of the verification procedure, which is presented in this paper, to a hydraulic tide model of Murrells Inlet, South Carolina, was quite successful and demonstrated that a reduction of required time for tidal model verification can be obtained by qualitative procedures of defining required model roughness and model areas requiring such roughness. The present procedure with its complex method of analyses requires the services of an ADACS for practical application within the time frame of typical hydraulic model studies. Although the methodology presented in this paper has shown much promise, additional model applications and laboratory experiments are needed to refine the application of the technique and to provide improved procedures for quantitative estimates of the horizontal distribution of model roughness in distorted, hydraulic tidal models.

7. SUMMARY. Application of ADACS to physical modeling techniques for tidal inlet studies has (1) reduced the required time for model verification and testing with a related cost reduction in model studies, (2) increased the quality and quantity of model data, and (3) allowed more sophisticated procedures for model control and data analyses with an improvement in information from model tests on which can be based engineering decisions regarding the planning and design of proposed channel dredging, inlet geometry changes, and the height, length, alignment, and orientation of coastal structures (e.g., jetties and breakwaters). Initial efforts in attempting to quantify the artificial simulation of prototype frictional effects in geometrically distorted hydraulic tidal models have shown much promise in the first application of the relative phase difference procedure, reported in this paper, to the tidal model of Murrells Inlet, South Carolina. Additional refinements to this procedure are required, and future model applications and/or

laboratory studies are desirable to define such improvements. The automation efforts, described in this paper, of physical hydraulic models for wave and tidal inlet studies has been highly successful in improving modeling techniques, enhancing modeling capabilities, and increasing the efficiency of such procedures through time and cost savings. Proposed future efforts in model automation at WES include improved sensors, spectral wave generation, and an expansion of these automated capabilities to other model facilities.

#### REFERENCES

1. Durham, D. L. and H. C. Greer, III, "Automated Data Acquisition and Control Systems for Hydraulic Wave Models." Paper presented at 1975 Army Numerical Analysis Conference, St. Louis, Missouri, Feb 1975.\*
2. Whalin, R. W., Chatham, C. E., Durham, D. L., and Pickett, E. B., "A Case History of Los Angeles-Long Beach Harbors," ASCE Proc. of International Symposium on Ocean Wave Measurement and Analysis Vol 1, Sep 1974.
3. "Roughness Standards for Hydraulic Models, Report 1, Study of Finite Boundary Roughness in Rectangular Flumes," Tech. Memo. No. 2-364, U. S. Army Engineer Waterways Experiment Station, Vicksburg, Miss., 1953.
4. de Vries, M., Conference on "Selected Problems from the Theory of Simulation of Hydrodynamic Phenomena" at Jablouna, Jun 1969, Polish Academy of Sciences, Institute of Hydro-Engineering, Gdonsk.
5. Multer, Roger, "Artificial Roughness in Distorted Hydraulic Models of Estuaries," Unpublished Memorandum (HM) on Chesapeake Bay Study, U. S. Army Engineer Waterways Experiment Station, Vicksburg, Miss., 1975.
6. Schureman, Paul, "Manual of Harmonic Analyses and Prediction of Tides," Special Publication No. 98, U. S. Dept. of Commerce, 1958.

---

\*This article appears near the end of these Proceedings.



## OPTIMAL INSTRUMENTATION PLANNING USING AN LDLT FACTORIZATION

William S. Agee, Robert H. Turner, and Jerry L. Meyer  
Analysis and Computation Division  
National Range Operations Directorate  
US Army White Sands Missile Range  
White Sands Missile Range, NM 88002

**ABSTRACT.** Optimal instrumentation planning procedures have been developed at White Sands Missile Range (WSMR) for dovap, radar, and cinetheodolite instrumentation systems. These procedures use an Instrumentation Plan (IP) improvement algorithm. The IP improvement algorithm was originally based on an update procedure for the IP covariance matrix. The large number of available instrumentation sites require a great amount of CPU time and storage using this update procedure. In addition, numerical instabilities are sometimes encountered. Recently, we have redeveloped the IP improvement algorithm based on updating the IP information matrix. The special structure of the information matrix and use of an LDLT factorization of the information matrix have reduced the CPU time by a factor of three, greatly reduced the storage requirements and eliminated the numerical instabilities.

**1. INTRODUCTION.** Trajectory estimation is a fundamental part of the mission at White Sands Missile Range. In order to do a good job of trajectory estimation, it is absolutely essential to have a good instrumentation plan (IP). The primary instrumentation systems at WSMR are radar, dovap, and cinetheodolite. For each of these systems the instrumentation planning problem can be stated informally as: given a nominal trajectory of flight path select the instrument sites from among those available for the instrumentation system so as to meet the range users data requirements. The range users data requirements are usually stated in terms of the precision or accuracy of the trajectory estimates. In the past this instrumentation planning problem has been handled by the use of various ad hoc techniques. The resulting instrumentation plans were usually adequate to meet the range users requirements, but they were often wasteful in the sense that an equally good instrumentation plan could have been developed using fewer instrument sites.

Recently we have developed some techniques which formalize the instrumentation planning problem for the three primary instrumentation systems. Our techniques solve the following problem. Given a set of  $N$  time points  $t_i$  entirely covering the nominal flight path and the corresponding position vectors  $\bar{x}_i$ ,  $i=1, N$  to the flight path select a set of  $M$  instrument sites which minimize

$$C = \sum_{i=1}^N w_i \text{trcov}(\bar{x}_i) \quad w_i > 0$$

The quantity  $C$  may be interpreted as the weighted sum of error estimates which would result if measurements from the nominal flight path were processed. The  $w_i$  are a set of weights used to attach more importance to some trajectory

points than to others.  $\text{cov}(\bar{x}_i)$  is the  $3 \times 3$  covariance matrix of the errors in the estimate of the trajectory position vector  $\bar{x}_i$ .

The number of surveyed instrument sites for each of the instrumentation systems is approximately: (a) radar - 15, (b) dovap - 650, and (c) cinetheodolite - 230. It is obvious that the selection of M sites from the large number of available sites would usually be computationally prohibitive if a procedure of enumeration and examination of all possible combinations of M sites is used to achieve a global minimum of C. Rather than pursuing a global minimum, we have satisfied ourselves with obtaining a local minimum of C through the use of an instrumentation plan improvement algorithm.

#### Instrumentation Plan Improvement Algorithm

a. Given an arbitrary initial IP having M instruments construct a modified IP having M+1 instruments by adding the instrument site from those available in an instrumentation planning pool (IPP) which results in the greatest decrease of C.

b. Delete the instrument site from the modified IP which results in the smallest increase of C.

c. Repeat the exchange procedure between the IPP and the IP given in steps a. and b. until no farther improvement is possible.

The minimum achieved by the IP improvement algorithm is local in the sense that it is dependent on the initial IP with which the algorithm started.

The IPP is a set of instrument sites considered feasible to use for the geometry of the given flight path. The IPP is obtained from the set of all existing sites by placing some basic constraints on site selection. The considerations which go into these basic constraints are:

(1) Dovap:

- (a) Reference signal strength available at the receiver sites.
- (b) Antenna nulls at low and high elevation angles.

(2) Radar: Ground clutter at low elevation angles.

(3) Cinetheodolite:

- (a) Minimum image size readable on film.
- (b) Maximum tracking rates.
- (c) Sun angle.
- (d) Flight safety evacuation area.

The effect of the addition and deletion of instrument sites on the objective function  $C$  required in steps a. and b. of the IP improvement algorithm requires a great deal of computation even though the number of sites to be considered has been reduced through the use of basic constraints in forming IIP. Each time a station is added or deleted the entire  $3N \times 3N$  IP covariance matrix must be updated. In a previous approach to updating the IP covariance matrix a direct update was used. The use of this direct update required excessive computation time. Also, severe numerical instability was encountered when using this update for radar site selection. Because of these problems the direct update was discarded in favor of a method which updates the IP information matrix. In all cases the IP information matrix has a special, sparse structure therefore requiring much less computation for an update. The numerical instabilities are also eliminated using this update procedure.

The remainder of the paper is devoted to describing the implementation of the IP improvement algorithm using the information matrix update. First, measurement models for the instrumentation systems will be described. Following this, the batch processors used to estimate the trajectory and their information and covariance matrices will be briefly described. Finally, the numerical procedures for updating the information matrices will be given.

2. MEASUREMENT MODELS. Each dovap receiver measures a loop range change between successive trajectory points. Thus, the measurement function for the  $\alpha$ th receiver is a function of the position vector  $\bar{x}_i$  and  $\bar{x}_{i+1}$  to the trajectory. Denote the measurement function by  $g_\alpha(\bar{x}_i, \bar{x}_{i+1})$ . The dovap observation, for the  $i$ th trajectory interval, denoted by  $Z_\alpha(t_i, t_{i+1})$  is

$$Z_\alpha(t_i, t_{i+1}) = g_\alpha(\bar{x}_i, \bar{x}_{i+1}) + n_\alpha(t_{i+1}) \quad (1)$$

where  $n_\alpha(t_{i+1})$  is a zero mean measurement noise term having variance  $R_\alpha(i+1)$ .

Let  $g_\alpha(\bar{x}_i)$  be a  $p$ -vector of measurement functions for the  $\alpha$ th radar or cine site at the  $i$ th trajectory point. Then the  $p$ -vector of radar or cine observations at the  $i$ th trajectory time is

$$Z_\alpha(t_i) = g_\alpha(\bar{x}_i) + b_\alpha + n_\alpha(t_i) \quad (2)$$

where  $b_\alpha$  is  $p$ -vector of zero-set measurement biases for the  $\alpha$ th site and  $n_\alpha(t_i)$  is a zero-mean measurement noise vector having diagonal covariance  $R_\alpha(i)$ . For cine measurements  $p=2$  and the components of  $g_\alpha(\cdot)$  are azimuth and elevation angles. For radar  $p=3$  and the components of  $g_\alpha(\cdot)$  are range, azimuth, and elevation.

3. DOVAP BATCH PROCESSOR AND INFORMATION MATRIX UPDATE. The dovap batch processor minimizes

$$Q(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_N) = \sum_{i=2}^N \sum_{\alpha=1}^M \frac{1}{R_\alpha(i)} (Z_\alpha(t_{i-1}, t_i) - g_\alpha(\bar{x}_{i-1}, \bar{x}_i))^2 \quad (3)$$

The information matrix of this batch processor is a  $3N \times 3N$  block tridiagonal matrix

$$A = \begin{bmatrix} A_1 & A_{12} & & & \\ A_{12}^T & A_2 & A_{23} & & \\ & A_{23}^T & A_3 & & \\ & & & \ddots & \\ & & & A_{N-1} & A_{N-1,N} \\ & & & & A_{N-1,N}^T & A_N \end{bmatrix} \quad (4)$$

where the  $A_i$ 's and  $A_{ij}$ 's are  $3 \times 3$  and are functions of the trajectory position vectors, the measurement noise variances, and the partial derivative vectors,  $G_{\alpha 1}(\bar{x}_i) = \partial g_{\alpha}(\bar{x}_i, \bar{x}_{i+1}) / \partial \bar{x}_i$  and  $G_{\alpha 2}(\bar{x}_{i+1}) = \partial g_{\alpha}(\bar{x}_i, \bar{x}_{i+1}) / \partial \bar{x}_{i+1}$ . The information matrix  $A$  is factored as  $A = LDL^T$  where  $L$  is unit lower triangular and  $D$  is diagonal. The special structure of  $A$  allows  $L$  to be chosen as

$$L = \begin{bmatrix} L_1 & & & & \\ L_{21} & L_2 & & & \\ & L_{32} & L_3 & & \\ & & & \ddots & \\ & & & & L_{N,N-1} & L_N \end{bmatrix} \quad (5)$$

where the  $L_i$  are  $3 \times 3$  unit lower triangular and the  $L_{ij}$  are  $3 \times 3$ . The matrix  $D$  is partitioned into  $3 \times 3$  diagonal blocks  $D_i$ . The  $L_i$ ,  $L_{ij}$ , and  $D_i$  are computed from

$$L_1 D_1 L_1^T = A_1 \quad (6a)$$

$$L_i D_i L_i^T = A_i - L_{i,i-1} D_{i-1} L_{i,i-1}^T \quad (6b)$$

$$L_i D_i L_{i+1,i}^T = A_{i,i+1} \quad (6c)$$

Adding or deleting the  $\beta^{\text{th}}$  receiver site during the  $K^{\text{th}}$  observation interval yields the information matrix update equation

$$A_{\pm} = A \pm J_{\beta}(\bar{x}_{K-1}, \bar{x}_K) J_{\beta}^T(\bar{x}_{K-1}, \bar{x}_K) \quad (7)$$

where  $A_{\pm}$  is the information matrix after adding a measurement (+) or deleting a measurement (-).  $J_{\beta}(\bar{x}_{K-1}, \bar{x}_K)$  is a  $3N$ -vector of the form

$$J_{\beta}^T(\bar{x}_{K-1}, \bar{x}_K) = [0 \ 0 \ - \ - \ - \ 0 G_{\beta 1}(\bar{x}_{K-1}) \ G_{\beta 2}(\bar{x}_K) 0 \ - \ - \ - \ 0] \quad (8)$$

(K-1)<sup>st</sup>
K<sup>th</sup>  
block
block

$J_{\beta}^T(\bar{x}_{K-1}, \bar{x}_K)$  is partitioned into 3-vectors with the only nonzero entries being in the (K-1)<sup>st</sup> and K<sup>th</sup> positions. If  $A$  is factored as  $L^{\pm} D^{\pm} L^{\pm T}$  then

$$L^{\pm} D^{\pm} L^{\pm T} = LDL^T + J_{\beta}(\bar{x}_{K-1}, \bar{x}_K) J_{\beta}^T(\bar{x}_{K-1}, \bar{x}_K) \quad (9)$$

Equating block elements on both sides of the last equation gives the update equations for  $L_i$ ,  $L_{ij}$ , and  $D_i$  as

$$L_i^{\pm} = L_i \quad \left. \begin{array}{l} L_i^{\pm} = L_i \\ D_i^{\pm} = D_i \\ L_{i+1,i}^{\pm} = L_{i+1,i} \end{array} \right\} i < K-1 \quad (10a)$$

$$D_i^{\pm} = D_i \quad (10b)$$

$$L_{i+1,i}^{\pm} = L_{i+1,i} \quad (10c)$$

$$L_{K-1}^{\pm} D_{K-1}^{\pm} L_{K-1}^{\pm T} = L_{K-1} D_{K-1} L_{K-1}^T \pm G_{\beta 1}^T(\bar{x}_{K-1}) G_{\beta 1}(\bar{x}_{K-1}) \quad (11)$$

$$L_{K-1}^{\pm} D_{K-1}^{\pm} L_{K,K-1}^{\pm T} = L_{K-1} D_{K-1} L_{K,K-1}^T \pm G_{\beta 1}^T(\bar{x}_{K-1}) G_{\beta 2}(\bar{x}_K) \quad (12)$$

$$L_K^{\pm} D_K^{\pm} L_K^{\pm T} = L_K D_K L_K^T + L_{K,K-1} D_{K-1} L_{K,K-1}^T - L_{K,K-1}^{\pm} D_{K-1}^{\pm} L_{K,K-1}^{\pm T} \pm G_{\beta 2}^T(\bar{x}_K) G_{\beta 2}(\bar{x}_K) \quad (13)$$

$$L_i^{\pm} D_i^{\pm} L_{i+1,i}^{\pm T} = L_i D_i L_{i+1,i}^T \quad K \leq i \leq N-1 \quad (14)$$

$$L_i^{\pm} D_i^{\pm} L_i^{\pm T} = L_i D_i L_i^T + L_{i,i-1} D_{i-1} L_{i,i-1}^T - L_{i,i-1}^{\pm} D_{i-1}^{\pm} L_{i,i-1}^{\pm T} \quad K < i \leq N \quad (15)$$

Some manipulation of the above equations reduces (13) and (15) to a rank one modification form so that the update equations become

$$\left. \begin{array}{l} L_i^{\pm} = L_i \\ D_i^{\pm} = D_i \\ L_{i+1,i}^{\pm} = L_{i+1,i} \end{array} \right\} i < K-1$$

$$L_{K-1}^{\pm} D_{K-1}^{\pm} L_{K-1}^{\pm T} = L_{K-1} D_{K-1} L_{K-1}^T \pm G_{\beta 1}^T (\bar{x}_{K-1}) G_{\beta 1} (\bar{x}_{K-1}) \quad (11)$$

$$L_{K-1}^{\pm} D_{K-1}^{\pm} L_{K,K-1}^{\pm T} = L_{K-1} D_{K-1} L_{K,K-1}^T \pm G_{\beta 1}^T (\bar{x}_{K-1}) G_{\beta 2} (\bar{x}_K) \quad (12)$$

$$L_K^{\pm} D_K^{\pm} L_K^{\pm T} = L_K D_K L_K^T \pm x_K x_K^T \quad (16)$$

with

$$x_K^T = (G_{\beta 2} (\bar{x}_K) - L_{K,K-1} L_{K-1}^{-1} G_{\beta 1} (\bar{x}_{K-1})) / (1 + q_{K-1})^{1/2} \quad (17)$$

$$q_{K-1} = G_{\beta 1} (\bar{x}_{K-1}) (L_{K-1} D_{K-1} L_{K-1}^T)^{-1} G_{\beta 1}^T (\bar{x}_{K-1}) \quad (18)$$

$$L_{i,i+1,i}^{\pm} D_{i,i+1,i}^{\pm T} = L_{i,i} D_{i,i} L_{i,i+1,i}^T \quad K \leq i \leq N-1 \quad (19)$$

$$L_{i,i,i}^{\pm} D_{i,i,i}^{\pm T} = L_{i,i} D_{i,i} L_{i,i}^T \pm x_i x_i^T \quad K \leq i \leq N \quad (20)$$

with

$$x_i^T = L_{i,i-1} L_{i-1}^{-1} x_{i-1}^T / (1 + q_{i-1})^{1/2} \quad (21)$$

$$q_{i-1} = x_{i-1} (L_{i-1} D_{i-1} L_{i-1}^T)^{-1} x_{i-1}^T \quad (22)$$

Equations (11), (16), and (20) are solved for the updated factors using methods c1 and c2 of Gill, Golub, Murray, and Sanders [1]. Method c1 is used for addition (+) because there are many more adds than deletes and c1 is the most computationally efficient of the update methods. Method c2 is used for delete (-) since stability can be a problem in this case. Equations (12) and (19) are solved for the rows of  $L_{i+1,i}^{\pm}$ .

After applying the above update equations sequentially to add or delete the  $\beta$ th receiver from all observation intervals, the effect of the  $\beta$ th receiver on the objective function C must be computed. The new objective function C is easily computed by obtaining the new values of  $\text{cov}(\bar{x}_i)$  from the available L and D factors of the information matrix.

$$\text{cov}(\bar{x}_N) = L_N^{-T} D_N^{-1} L_N^{-1} \quad (23)$$

$$\text{cov}(\bar{x}_{i-1}) = L_{i-1}^{-T} (L_{i,i-1}^T \text{cov}(\bar{x}_i) L_{i,i-1} + D_{i-1}^{-1}) L_{i-1}^{-1} \quad i=2, N \quad (24)$$

4. RADAR AND CINE BATCH PROCESSOR AND INFORMATION MATRIX UPDATE. The radar and cine batch processors minimize

$$Q(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_N) = \sum_{i=1}^N \sum_{\alpha=1}^M (Z_{\alpha}(t_i) - g_{\alpha}(\bar{x}_i) - S_{\alpha}b)^T R_{\alpha}^{-1}(i) (Z_{\alpha}(t_i) - g_{\alpha}(\bar{x}_i) - S_{\alpha}b) \quad (25)$$

where  $b$  is the composite bias vector

$$b^T = [b_1 b_2 \dots b_M] \quad (26)$$

and

$$S_{\alpha}b = b_{\alpha} \quad (27)$$

The information matrix of this batch processor is a  $(3N+pM) \times (3N+pM)$  bordered block diagonal matrix

$$A = \begin{bmatrix} A_1 & & & A_{1,N+1} \\ & A_2 & & A_{2,N+1} \\ & & \ddots & \vdots \\ & & & A_N & A_{N,N+1} \\ A_{1,N+1}^T & A_{2,N+1}^T & \dots & A_{N,N+1}^T & A_{N+1} \end{bmatrix} \quad (28)$$

The  $A_i$   $i=1, N$  are  $3 \times 3$ ,  $A_{N+1}$  is  $pM \times pM$  and the  $A_{i,N+1}$  are  $3 \times pM$ . These matrices are functions of the trajectory position vectors, the measurement noise variances and the partial derivative matrices  $\partial g_{\alpha}(\bar{x}_i) / \partial \bar{x}_i = G_{\alpha}(\bar{x}_i)$ .  $A$  is factored as  $A = LDL^T$  where  $L$  is unit lower triangular and  $D$  is diagonal. The simple structure of  $A$  allows  $L$  to be chosen as

$$L = \begin{bmatrix} L_1 & & & & \\ & L_2 & & & \\ & & \bigcirc & & \\ & & & \ddots & \\ & & & & L_N \\ L_{N+1,1} & L_{N+1,2} & & & L_{N+1,N} & L_{N+1} \end{bmatrix} \quad (29)$$

The  $L_i$  are  $3 \times 3$  unit lower triangular,  $L_{N+1,i}$  are  $p \times 3$ , and  $L_{N+1}$  is  $p \times p$  unit lower triangular. The matrix  $D$  is partitioned into  $N$   $3 \times 3$  diagonal blocks  $D_i$  and one  $p \times p$  diagonal block  $D_{N+1}$ . The  $L_i$ ,  $L_{N+1,i}$ , and  $D_i$  are computed from

$$L_i D_i L_i^T = A_i \quad i=1, N \quad (30a)$$

$$L_i D_i L_{N+1,i}^T = A_{i,N+1} \quad (30b)$$

$$L_{N+1} D_{N+1} L_{N+1}^T = A_{N+1} - \sum_{i=1}^N L_{N+1,i} D_i L_{N+1,i}^T \quad (30c)$$

Adding or deleting the  $\beta$ th instrument site at the  $K^{\text{th}}$  time point yields the information matrix update equation

$$A_{\pm} = A \pm J_{\beta}(\bar{x}_K) J_{\beta}^T(\bar{x}_K) \quad (31)$$

where  $A_{\pm}$  is the information matrix after adding a measurement (+) or deleting a measurement (-).  $J_{\beta}(\bar{x}_K)$  is a  $(3N+pM) \times p$  matrix which is partitioned as

$$J_{\beta}^T(\bar{x}_K) = \begin{bmatrix} 0 & \cdots & 0 & G_{\beta}(\bar{x}_K) & 0 & \cdots & 0 & S_{\beta} \end{bmatrix} \quad (32)$$

$K^{\text{th}}$ 
 $(N+1)^{\text{st}}$   
block
block

If  $A_{\pm}$  is factored as  $L^{\pm} D^{\pm} L^{\pm T}$

$$L^{\pm} D^{\pm} L^{\pm T} = L D L^T + J_{\beta}(\bar{x}_K) J_{\beta}^T(\bar{x}_K) \quad (33)$$

Equating block elements on both sides of the last equation gives the update equations for  $L_i$ ,  $L_{N+1,i}$ , and  $D_i$  as



$$\left. \begin{aligned} L_i^\pm &= L_i \end{aligned} \right\} \quad (34a)$$

$$\left. \begin{aligned} D_i^\pm &= D_i \end{aligned} \right\} \quad i \neq K \quad (34b)$$

$$\left. \begin{aligned} L_{N+1,i}^\pm &= L_{N+1,i} \end{aligned} \right\} \quad (34c)$$

$$L_K^\pm D_K^\pm L_K^{\pm T} = L_K D_K L_K^T \pm G_\beta^T(\bar{x}_K) G_\beta(\bar{x}_K) \quad (35)$$

$$L_K^\pm D_K^\pm L_{N+1,K}^{\pm T} = L_K D_K L_{N+1,K}^T \pm G_\beta^T(\bar{x}_K) S_\beta \quad (36)$$

$$\begin{aligned} L_{N+1}^\pm D_{N+1}^\pm L_{N+1}^{\pm T} &= L_{N+1} D_{N+1} L_{N+1}^T + L_{N+1,K} D_K L_{N+1,K}^T \\ &\quad - L_{N+1,K}^\pm D_K^\pm L_{N+1,K}^{\pm T} \pm S_\beta^T S_\beta \end{aligned} \quad (37)$$

The above update equations are solved by reducing (35) and (37) to a sequence of p rank one modifications. Let  $G_{\beta j}^T(\bar{x}_K)$  be the jth column of  $G_\beta^T(\bar{x}_K)$  and  $S_{\beta j}^T$  the jth column of  $S_\beta^T$ . Then (35)-(37) reduce to

$$L_K^\pm D_K^\pm L_K^{\pm T} = L_K D_K L_K^T \pm G_{\beta j}^T(\bar{x}_K) G_{\beta j}(\bar{x}_K) \quad (38)$$

$$L_K^\pm D_K^\pm L_{N+1,K}^{\pm T} = L_K D_K L_{N+1,K}^T \pm G_{\beta j}^T(\bar{x}_K) S_{\beta j} \quad (39)$$

$$L_{N+1}^\pm D_{N+1}^\pm L_{N+1}^{\pm T} = L_{N+1} D_{N+1} L_{N+1}^T \pm x_j^T x_j \quad j=1,p \quad (40)$$

$$\text{with } x_j^T = (S_{\beta j}^T - L_{N+1,K} L_K^{-1} G_{\beta j}^T(\bar{x}_K)) / (1 \pm q)^{1/2} \quad (41)$$

$$q = G_{\beta j}(\bar{x}_K) (L_K D_K L_K^T)^{-1} G_{\beta j}^T(\bar{x}_K) \quad (42)$$

In the above  $L_K \leftarrow L_K^\pm$ ,  $D_K \leftarrow D_K^\pm$ ,  $L_{N+1,K} \leftarrow L_{N+1,K}^\pm$ ,  $L_{N+1} \leftarrow L_{N+1}^\pm$  after the jth rank one modification. Methods c1 and c2 of [1] are again used to solve for the updated factors.

After applying the above update equations sequentially to add or delete the  $\beta$ th instrument site at all trajectory time points, the effect of the  $\beta$ th instrument on the objective function C must be computed. The new objective function C is easily computed by obtaining the new values of  $\text{cov}(\bar{x}_j)$  from the available L and D factors of the information matrix.

$$L_{N+1} P_i = L_{N+1,i} \quad (42)$$

$$\text{cov}(\bar{x}_i) = L_i^{-T} (D_i^{-1} + P_i^T D_{N+1}^{-1} P_i) L_i^{-1} \quad i=1, N \quad (43)$$

#### REFERENCES

1. Gill, P. E., G. H. Golub, W. Murray, and M. A. Sanders; "Methods for Modifying Matrix Factorizations"; Math of Comp, Vol 28, No. 126, p 505-535, April 1974.

A COMPUTER SOLUTION OF THE BUCKINGHAM PI THEOREM  
USING SYMBOLANG, A SYMBOLIC MANIPULATION LANGUAGE\*

Morton A. Hirschberg  
US Army Ballistic Research Laboratories  
Aberdeen Proving Ground, Maryland 21005

ABSTRACT

Among the theories of similitude, application of the Buckingham Pi Theorem allows one to find meaningful relationships among variables, check the formulation of a system of equations, and allow prediction from scaled parameters. Simply stated, the Pi Theorem asserts that if there are  $n$  variables involving  $N$  fundamental units, these may be combined to form  $n-N$  dimensionless parameters each containing  $N+1$  variables.

SYMBOLANG is a FORTRAN-based symbol manipulator using a list structure. Symbol manipulators operate on strings of symbols rather than numbers. As an example, a symbol manipulator allows multiplication of  $N+1$  by  $N-1$  to obtain  $N^2-1$ . A list structure allows data to be stored and manipulated by relationships rather than sequentially or by some other scheme.

The solution of the Pi Theorem involves forming Pi Terms. This is done by the investigator ordering the equations of the system in order of importance. The equations are then exponentiated and multiplied together in groups depending upon the sizes of  $n$  and  $N$ . Once a Pi Term is formed, the dimensionless set is solved for by setting the exponents of the parameters to zero and using any routine which will solve a linear set of equations. The numeric solution is then paired with the parameter to which it belongs.

Planned extensions of this work involve tabling well known dimensionless numbers and checking generated solutions against the tabled set. Furthermore, data for the parameters can be input and full numeric answers obtained. In addition, the computer can generate a large number of solutions by permuting the equations. Permuted solutions can be factor-analyzed and a regression fit made to determine a "best-solution."

---

\*This article appeared as BRL Report No. 1824.

*The method of dimensions is somewhat dangerous but when used with proper care, it is unquestionably of great power and value.*<sup>1</sup>

## I. INTRODUCTION

The theory of dimensions has a long history of development dating back to the ancient Greeks.<sup>2</sup> Galileo, Newton, Fourier, Lord Rayleigh, Vaschy, Buckingham, and Bridgeman are a few of the men instrumental in the development of dimensional analysis.

The theory of dimensions is important when we wish to compare results for two arbitrarily selected systems of units. It is useful in areas where knowledge is developing through an intermediate stage, when basic laws are already known but problem solutions are lacking. Its application may yield interesting conclusions based upon general physical assumptions which are themselves uninteresting.<sup>3</sup>

The theory of dimensions has three important applications<sup>4</sup>:

1. It supplies one with useful checks against errors made in calculations.
2. It suggests forms of physical laws.
3. It allows the prediction of behavior of a full-scale system from the behavior of a model.

The rest of this paper will deal with all three applications; however, the very nature of the method employed guarantees the first application, and planned extensions of the model will form the basis for a future report of the third application.

---

<sup>1</sup>Strutt, J.W., Baron Rayleigh, The Theory of Sound, Vol. I, New York, Dover Publications, 1945.

<sup>2</sup>Macagno, E.O., Historico-Critical Review of Dimensional Analysis, Journal of the Franklin Institute, 292, 391-402, 1971.

<sup>3</sup>Sedov, L.I., Similarity and Dimensional Methods in Mechanics, New York, Academic Press, 1959.

<sup>4</sup>Synge, J.L., & Griffith, B.A., Principles of Mechanics, New York, McGraw-Hill, 1942.

## II. THE BUCKINGHAM PI THEOREM

In dealing with the equations of physical phenomena, one often multiplies and divides various quantities and/or parameters. The rules of combination are essentially inherent in the parameters at hand. If two physical quantities possess the same dimensionality, we say their quotient is dimensionless or non-dimensional. Pi terms are the names given to such non-dimensional products or quotients of the original parameters.<sup>5</sup>

As a general rule, it takes  $N+1$  variables to form a dimensionless term with  $N$  fundamental units. This leads one to a simple statement of the Pi Theorem:

If there are  $n$  variables involving  $N$  fundamental units, they can be combined to form  $n-N$  dimensionless parameters, each containing  $N+1$  variables.<sup>6</sup>

The Pi Theorem was first proved by Buckingham<sup>7</sup> and is often referred to as the Buckingham Pi Theorem.

As an example, all the quantities in mechanics can be expressed using three fundamental units, e.g. force (F), length (L), and time (T). All quantities in mechanics can be expressed in the form  $F^\alpha L^\beta T^\gamma$  where  $\alpha$ ,  $\beta$ , and  $\gamma$  are positive, zero, or negative powers which are not necessarily integers. The following is a list of a few of the dimensions which can be formed:

---

<sup>5</sup>Baker, W.E., Westine, P.S., & Dodge, F.T., Similarity Methods in Engineering Dynamics: Theory and Practice of Scale Modeling, New Jersey, Spartan Books, 1973.

<sup>6</sup>Housner, C.W., & Hudson, D.E., Applied Mechanics Dynamics, New York, van Nostrand, 1950.

<sup>7</sup>Buckingham, E., On Physically Similar Systems: Illustrations of the Use of Dimensional Equations, Physical Review, 2, 345-376, 1914.

[Velocity] =  $LT^{-1}$   
 [Acceleration] =  $LT^{-2}$   
 [Force] =  $F$   
 [Moment of a Force] =  $FL$   
 [Linear momentum] =  $FT$   
 [Angular momentum] =  $FLT$   
 [Density] =  $FT^2 L^{-4}$   
 [Power] =  $FLT^{-1}$   
 [Pressure] =  $FL^{-2}$   
 [Viscosity] =  $FL^{-2} T$

where the "=" is a true equality.

The solution of a dimensional set of equations involves forming Pi Terms and solving the system of equations formed by the resulting coefficients of Pi Terms. That is, one solves equations of the form

$$C_{i1} + C_{i2}\alpha + C_{i3}\beta + C_{i4}\gamma = 0 \quad i = 1, 2, 3$$

where each equation is the coefficient of L, F, and T respectively. One solution is formed for each possible Pi Term.

A computer program, described in a later section, has been developed to form the Pi Terms and solve the resulting equations which render the system dimensionless.

### III. FORMULA MANIPULATION

Formula manipulation (or symbolic manipulation) is primarily processing of non-numeric data. Several languages exist for such purposes.<sup>8</sup> Basically, a formula manipulator is written as a specific list processing language, or language which processes strings of symbols.

---

<sup>8</sup>Sammet, J.L., Survey of Formula Manipulation, Communications of the Association for Computing Machinery, 9, 555-569, 1966.

The language selected to form Pi Terms and solve the system of coefficient equations was SYMBOLANG.<sup>9,10</sup> SYMBOLANG is written almost completely in FORTRAN and is itself an extension of SLIP,<sup>11,12,13</sup> a list-processing language also written in FORTRAN.

The combined SYMBOLANG-SLIP system comprises nearly 180 sub-programs and allows for arithmetic operations (addition, subtraction, multiplication, division), exponentiations, substitutions, evaluations, and differentiations.

Lists may be created, manipulated, and destroyed, with storage of destroyed lists being reusable. In addition, recursive or repetitive processing is permitted in a limited sense (using the computed GO TO statement). In toto, the SYMBOLANG-SLIP system is powerful and easily extendable. No new language is required to use the system. One need be familiar with FORTRAN and acquainted with the subprograms comprising SYMBOLANG and SLIP.

#### IV. METHOD OF SOLUTION

The number of parameters and the parameters of the system are input and displayed. This is done for completeness. Next, the number of equations,  $n$ , and the equations of the system are input. The latter are input as SYMBOLANG lists and are assumed to be ordered with those equations deemed most important first. That is, earlier entered equations are more likely to produce fruitful results. The equations are displayed. The number of fundamental units,  $N$ , is also an input. From  $n$  and  $N$ , the number of Pi Terms to be produced is calculated.

The equations are then partitioned into new SYMBOLANG lists (herein called factors) containing the portion to the right of the equal sign. Similarly, the portion to the left of the equation is also retained (this will now be referred to as a variable). Factors involve fundamental units of the system.

---

<sup>9</sup>Finder, N.V., Pfaltz, J.L., & Bernstein, H.J., Four High-Level Extensions of FORTRAN IV: SLIP, AMPPL-II, TREETRAN, SYMBOLANG, New York, Spartan Books, 305-387, 1972.

<sup>10</sup>Hirschberg, M.A., SYMBOLANG-A SLIP Extension for Algebraic Manipulation, Ballistic Research Laboratories Report No. 1749, Nov 1974. (AD #A003190)

<sup>11</sup>Weizenbaum, J., Symmetric List Processor, Communications of the Association for Computing Machinery, 6, 524-544, 1963.

<sup>12</sup>Reference 9, pp 1-82.

<sup>13</sup>Hirschberg, M.A., SLIP for the BRLESC II COMPUTER, Ballistic Research Laboratories Report No. 1731, July 1974. (AD #A000653)

Pi Terms are formed by systematically raising each newly created factor to a power (powers are expressed as variables for solution, that is A, B, C, etc.) and then multiplying the exponentiated factors together.

There is always one factor in each Pi Term that has as its exponent 1. That is, it is not raised to a power, but is used as is, while all the other factors are exponentiated. This selection of primary factors is made systematically.

Once the product term has been formed (all exponentiated factors multiplied together), the product is displayed. It is of the form

$$\text{FACTOR}_1^{\alpha_1} \text{FACTOR}_2^{\alpha_2} \text{FACTOR}_3^{\alpha_3} \dots \text{FACTOR}_n^{\alpha_n}$$

$$\text{where } \alpha_i = C_{1i} A + C_{2i} B + C_{3i} C \dots$$

and  $C_{ji}$  is a signed integer 0,  $\pm 1$ ,  $\pm 2$ ,  $\pm 3$ , etc.

The equations are solved by setting the  $\alpha_i$ 's to zero and then evaluating the resulting set of simultaneous linear equations (this can be done using any matrix inversion routine where the solution vector is calculated).

The solutions of the simultaneous set of equations are then paired with their proper variable. This completes both the symbolic and numerical portions of the solution of the Pi Theorem.

Appendix I contains the computer listings for solving the Buckingham Pi Theorem.

## V. A SAMPLE PROBLEM

The following problem is adapted from Housner and Hudson. Let us consider the drag force  $F$  acting upon a body moving through a fluid. Let us assume a constant velocity  $v$  through a fluid of density  $\rho$  and viscosity  $\mu$ . The analysis is to apply to bodies of a specified shape, so the cross-sectional area  $A$  may be used as a measure of the body's size. The following variables and fundamental units enter into the problem:



Variable		Fundamental Unit
F	=	F
$\mu$	=	$FL^{-2} T$
A	=	$L^2$
$\rho$	=	$FL^{-4} T^2$
v	=	$LT^{-1}$

Two Pi Terms can be formed from this set. Using Force and Viscosity as the unique variables for solution, we arrive at

$$\pi_1 = F A^A \rho^B v^C \quad \text{and} \quad \pi_2 = \mu A^A \rho^B v^C.$$

Expanding these equations in terms of the fundamental units and solving the exponential equations, we arrive at the following solution:

$$\pi_1 = \frac{F}{A \rho v^2} \quad \text{and} \quad \pi_2 = \frac{\mu}{v A^{1/2} \rho}.$$

Table I shows the computer inputs and outputs for the problem above. Each Pi Term is developed and displayed. Following the Pi Term is the numerical solution of the exponents giving rise to the dimensionless set.

## VI. DISCUSSION

The presented computer solution of the Buckingham Pi Theorem provides one solution for a physical system. It is not necessarily an optimum or "best" solution.<sup>14,15</sup> Similarly the computer solution

<sup>14</sup>Blau, G.E., Optimization of Models Derived by Dimensional Analysis Using Generalized Polynomial Programming, Journal of the Franklin Institute, 292, 519-526, 1971.

<sup>15</sup>Chen, W.K., Algebraic Theory of Dimensional Analysis, Journal of the Franklin Institute, 292, 403-422, 1971.

TABLE I  
INPUTS AND OUTPUTS FOR THE  
COMPUTER SOLUTION OF THE BUCKINGHAM PI THEOREM

NUMBER OF PRIMITIVES	9	PI THEOREM SOLVER
----------------------	---	-------------------

PRIMITIVES ONLY USED FOR COMPLETENESS

FORCED\$ VISCOSITY\$ AREA\$ DENSITY\$ VELOCITY\$ FORCE\$ LENGTH\$ TIME\$	}	INPUTS
-----------------------------------------------------------------------------------------------	---	--------

THERE ARE 5 FORMULAS INVOLVING 3 VARIABLES

FORCED=FORCE\$ VISCOSITY=FORCE*TIME/LENGTH**2\$ AREA=LENGTH**2\$ DENSITY=FORCE*TIME**2/LENGTH**4\$ VELOCITY=LENGTH/TIME\$	}	INPUTS
---------------------------------------------------------------------------------------------------------------------------------------	---	--------

THERE ARE 2 PI TERMS

$LTV = FORCE^{(1 + B)*LENGTH^{(2*A - 4*B + C)} * TIME^{(2*B - C)}}$	}	OUTPUTS
---------------------------------------------------------------------	---	---------

\$

SEND OF EXPRESSION

SOLUTION FOR PI TERM

FORCED     ** .10000000E 01 AREA       ** -.10000000E 01 DENSITY    ** -.10000000E 01 VELOCITY   ** -.20000000E 01	}	OUTPUTS
-----------------------------------------------------------------------------------------------------------------------------	---	---------

$LTV = FORCE^{(1 + B)*LENGTH^{(-2 + 2*A - 4*B + C)*TIME^{(1 + 2*B - C)}}$	}	OUTPUTS
---------------------------------------------------------------------------	---	---------

\$

SEND OF EXPRESSION

SOLUTION FOR PI TERM

VISCOSITY   ** .10000000E 01 AREA        ** -.50000000E 00 DENSITY     ** -.10000000E 01 VELOCITY    ** -.10000000E 01	}	OUTPUTS
---------------------------------------------------------------------------------------------------------------------------------	---	---------

might need to have a transformation applied to it to put it in a form where it proves more usable (i.e. cubing the terms of one of the Pi Terms, etc.).

If the user's insight was good, the numerical solution (results obtained by using values for the parameters) one would obtain would be the same as those for an optimum or "best" solution.

The planned extension of this work is to provide data for the variables and calculate a Buckingham Pi Theorem solution. Next apply numerical values and save the results. Next calculate a new Buckingham Pi Theorem solution and again apply numerical values to this, saving the results. The computer can be programmed to do this for a great many of the total possible Pi solutions. The numerical results can then be factor-analyzed and a regression fit made. The numerical data can then be fed back through each Pi solution and solutions closest to the regression can be saved for later analysis. In this manner, a quasi-empirical best solution can be obtained for further study and use in later analysis of the system under consideration.

Furthermore, common non-dimensional numbers<sup>16</sup> may be tabled and as computed solutions are formed, these may be compared to the tabled values. If a match is found this can be noted, as it is of considerable use to know if a computed Pi Term is one of the common dimensionless numbers.

---

<sup>16</sup>Land, N.S., A Compilation of Non-dimensional Numbers, Washington, D.C., U.S. Government Printing Office, NASA SP-274, 1972.

## APPENDIX I

### Computer Listing of the Buckingham Pi Theorem Solution

```

PROGRAM RUCKY
C
C THIS IS THE PI THEOREM SOLVER
C WRITTEN BY MA HIRSCHBERG
C OCTOBER 1974
C
C SET UP SLIP STORAGE
COMMON AVSL,X(100)
DIMENSION SP(10000)
C SET UP PROGRAM STORAGE
DIMENSION LNEW(100),LTEMP(100),LPOWER(100)
DIMENSION LPRIMS(100),LFORMS(100)
DIMENSION SOLTN(24)
DIMENSION I7EE(78)
DIMENSION FFORM(100)
DIMENSION LEXP(78)
DATA ONE/1./
C DEFINE EXPONENTS A=7,A1=71,A2=72
DATA I7EF/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,1HL,1HM,1HN,
1 1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ,
22HA1,2HB1,2HC1,2HD1,2HE1,2HF1,2HG1,2HH1,2HI1,2HJ1,2HK1,2HL1,2HM1,
32HN1,2HO1,2HP1,2HQ1,2HR1,2HS1,2HT1,2HU1,2HV1,2HW1,2HX1,2HY1,2HZ1,
42HA2,2HB2,2HC2,2HD2,2HE2,2HF2,2HG2,2HH2,2HI2,2HJ2,2HK2,2HL2,2HM2,
52HN2,2HO2,2HP2,2HQ2,2HR2,2HS2,2HT2,2HU2,2HV2,2HW2,2HX2,2HY2,2HZ2/
C
C SETUP WORKING STORAGE
CALL INITAS(SP,10000)
C READ NUMBER OF PRIMITIVES
READ (5,10) NPRMS
10 FORMAT (15)
WRITE (6,12) NPRMS
12 FORMAT(1H1,55X,18H PI THEOREM SOLVER/
1 21H NUMBER OF PRIMITIVES,4X,15)
WRITE (6,14)
14 FORMAT (38H0PRIMITIVES ONLY USED FOR COMPLETENESS)
C READ PRIMITIVES
DO 20 I=1,NPRMS
LPRIMS(I)=INLIST(LPRIMS(I),5HINPUT,999)
20 CONTINUE
C OUTPUT PRIMITIVES
DO 30 I=1,NPRMS
CALL LSQNT(LPRIMS(I),10HPRIMITIVES,999.,TEMP)
30 CONTINUE
C
C READ NUMBER OF FORMULAS
READ (5,10) NFRMS
C READ NUMBER OF INDEPENDENT VARIABLES
READ (5,10) IVARS
WRITE (6,32) NFRMS,IVARS
32 FORMAT (10H0THERE ARE,1X,15,1X,8HFORMULAS,
1 1X,9HINVOLVING,1X,15,1X,9H VARIABLES)
C READ FORMULAS
DO 40 I=1,NFRMS
CALL INLIST(LFORMS(I),5HINPUT,3HVAL,TEMP)
40 CONTINUE

```

```

C
C CALCULATE THE NUMBER OF PI TERMS
  IPI=NFRMS-1VARS
  WRITE (6,42) IPI
  42 FORMAT (10HOTHER ARE,1X,15,1X,8HPI TERMS)
C
C OUTPUT FORMULAS
C   DO 50 I=1,NFRMS
C     CALL LSQNT(LFORMS(I),8HFORMULAS,999.,TEMP)
C   50 CONTINUE
C
C STRIP EQUALS OFF AND SETUP NEW SYMBOLANG LISTS
  IXDT=0
  DO 200 I=1,NFRMS

C
C SET UP READER FOR FORMULAS
  LRD=LRDRDV(LFORMS(I))
C
C STRIP FIRST PART OF FORMULA
  DO 60 J=1,4
    DATUM=ADVSR(LRD,FLAG)
    IF (J,NF,3) GO TO 60
    IXDT=IXDT+1
C
C SAVE VARIABLE NAME IN FORMULA FOR LATER USE (SOLUTION)
  FFORM(IXDT)=DATUM
  60 CONTINUE
  LW=0
  LW=LIST(LW)
  IC=0
  65 CONTINUE
C
C ADVANCE THROUGH LIST
  DATUM=ADVSR(LRD,FLAG)
  IF (FLAG,NF,0.) GO TO 70
C
C SET UP TEMPORARY LIST AND COUNT ELEMENTS
  IC=IC+1
  CALL NEWBOT(DATUM,LW,TEMP)
  GO TO 65
  70 CONTINUE
C
C SET COUNT AND FORM NEW NEW SYMBOLANG LIST
  LNEW(I)=0
  LNEW(I)=LIST(LNEW(I))
  LRD=LRDRDV(LW)
  LTEMP(I)=LIST(9)
  CALL NEWBOT(LTEMP(I),LNEW(I),TEMP)
  LC=IC-2
  IF (LC,LE,0) GO TO 2000
  DO 80 J=1,LC
    DATUM=ADVSR(LRD,FLAG)
    CALL NEWBOT(DATUM,LTEMP(I),TEMP)
  80 CONTINUE
C
C ERASE TEMPORARY LIST AND PRINT NEW LIST
  CALL IRALST(LW)
C   CALL LSQNT(LNEW(I),4HLNEW,999.,TEMP)
  200 CONTINUE
  K=NFRMS-1

```

```

      DO 210 J=1,K
C
C  SETUP EXPONENTS
      LEXP(1)=LSQMN1(LSQMN3(1.,IZEE(1),1.))
210 CONTINUE
C
C  SETUP PI TERMS
      LFRMS=LSQMN1(LSQMN1(1.))
      DO 260 J=1,IPI
      LTV=LSQMN1(LSQMN1(1.))
      KK=0
C  CYCLE THROUGH FORMULAS MOST IMPORTANT AND SKIP FORMULAS SO AS
C  TO INCLUDE ONLY THE IMPORTANT FORMULAS ONE AT A TIME
      DO 245 J=1,NFRMS
      IF (J.EQ.1) GO TO 215
      IF (J+IVARS.LE.NFRMS) GO TO 245
      KK=KK+1
C
C  RAISE POWER
      LPOWER(KK)=LSQRAZ(LNEW(J),LEXP(KK))
      LTU=LSQMEX(LTV,LPOWER(KK))
      CALL LSQDES(LTV,TEMP)
      TLGD=SEDRDR(LTU)
      LTV=LSQCPY(TLGD)
      CALL LSQDES(LTV,TEMP)
      GO TO 245
215 CONTINUE
      CALL LSQDES(LTV,TEMP)
      LTV=LSQMEX(LFRMS,LNEW(J))
245 CONTINUE
      CALL LSQPNT(LTV,3HLT,999.,TEMP)
      CALL PRLSTS(LTV,4)
C
C  SOLVE EQUATIONS
      CALL RUCKSV(LTV,IZEE,SOLTN,ICOUNT)
      CALL LSQDES(LTV,TEMP)
      DO 250 J=1,KK
      CALL LSQDES(LPOWER(J),TEMP)
250 CONTINUE
C
C  PRINT SOLUTION FOR PI TERM
      WRITE (6,251)
251 FORMAT( 21H0SOLUTION FOR PI TERM)
      KK=0
      DO 258 J=1,NFRMS
      IF (J.EQ.1) WRITE (6,253) FFORM(1),ONE
253 FORMAT (1H ,A10,1X,2H** ,1X,E14.8)
      IF (J+IVARS.LE.NFRMS) GO TO 258
      KK=KK+1
      WRITE (6,253) FFORM(J),SOLTN(KK)
258 CONTINUE
260 CONTINUE
      CALL EXIT
2000 CONTINUE
      WRITE (6,2010)
2010 FORMAT (7H NO NO      )
      CALL EXIT
      END
      SUBROUTINE RUCKSV(LIST, ZEE,SOLTN,ICOUNT)
C

```

C THIS SUBROUTINE SETS UP THE SOLUTION FOR THE PI THEOREM  
 C WRITTEN BY MA HIRSCHBERG  
 C JANUARY 1975  
 C

DIMENSION ZEE(78)  
 DIMENSION SOLTN(24)  
 DIMENSION AMAT(24,24)  
 DIMENSION AWORD(3)

C  
 C CLEAR STORAGE

DO 5 I=1,24  
 SOLTN(I)=0.0  
 DO 4 J=1,24  
 AMAT(I,J)=0.0  
 4 CONTINUE  
 5 CONTINUE

C  
 C SET UP READER FOR LIST  
 LR=LRDROV(LIST)

C SET FLAGS  
 LEVEL=0  
 ICOUNT=1  
 IEND=0  
 IWORD=0  
 JGO=0  
 JEND=1  
 10 CONTINUE  
 JGO=JGO+1  
 JGO=0

C ADVANCE THROUGH LIST  
 X=ANSWR(LR,K)  
 IF (K) 100,20,100  
 20 IF (LEVEL=LCNTR(LR)) 150,30,70  
 30 IF (NAMTST(X)) 60,40,60  
 40 IF (LISTMT(X)) 50,10,50

C WE HIT A SUBLIST  
 50 CONTINUE  
 LEVEL=LEVEL+1  
 IEND=0  
 IWORD=0  
 GO TO 10

C WE HIT A DATUM ELEMENT  
 60 CONTINUE  
 IEND=0  
 IF (JGO,LE,3) GO TO 10  
 IF (JGO,EQ,1) ICOUNT=ICOUNT+1  
 IF (JGO,EQ,1) GO TO 10  
 IWORD=IWORD+1

C STORE DATUM  
 AWORD(IWORD)=X  
 GO TO 10

C WE HIT AN END OF SUBLIST  
 70 CONTINUE  
 LEVEL=LEVEL-1  
 IEND=IEND+1  
 IF (IEND,LT,2) GO TO 75  
 JGO=1  
 GO TO 20  
 75 CONTINUE  
 IF (IWORD,GT,1) GO TO 80



```

      IF (IWORD.LE.0) GO TO 20
C STORE NUMERICAL COEFFICIENT (CONSTANT TERM)
      SOLTN(ICOUNT)=-AWORD(1)
      IWORD=0
      GO TO 20
      80 CONTINUE
C STORE MATRIX COEFFICIENT
      DO 90 I=1,78
      IF (AWORD(2).NE. ZEE(I)) GO TO 90
      AMAT(ICOUNT,I)=AWORD(1)
      IWORD=0
      GO TO (20,160), JEND
      90 CONTINUE
      95 CONTINUE
      CALL SLPERR(10H BUCKSV )
      100 IF (LEVEL-ICNTR(LR)) 150,120,110
      110 CONTINUE
      LEVEL=LEVEL-1
      GO TO 100
      120 CONTINUE
      CALL RCELL(LR)
      150 CONTINUE
      JEND=2
      GO TO 80
      160 CONTINUE
C
C INVERT MATRIX TO FIND NUMERICAL SOLUTION
      CALL MATINV(AMAT,ICOUNT,SOLTN,24,1,DET)
      IF (DET.EQ.0.) GO TO 95
      RETURN
      END

```



**- PIPS -**

**AN INTERACTIVE GRAPHICS PROGRAM  
FOR DETERMINATION OF MASS  
PROPERTIES OF IRREGULAR PLANAR SOLIDS**

**R. I. ISAKOWER**

**AND**

**F. R. TEPPER**

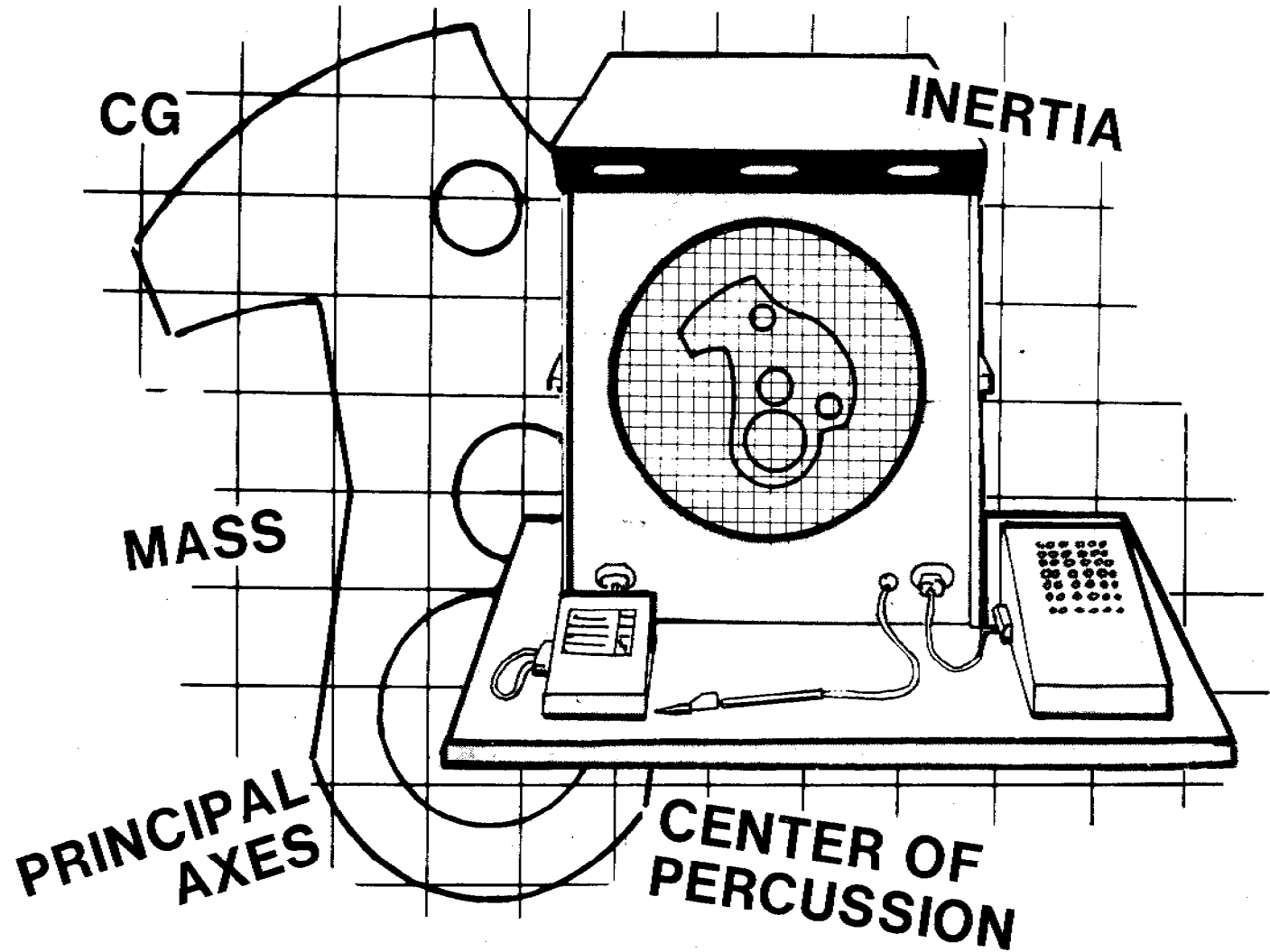
**PICATINNY ARSENAL, DOVER, N.J.**

**ABSTRACT**

PIPS is an interactive computer graphics program for the calculation of the mass properties of irregular planar solids. This program utilizes the graphics terminal to determine the mass, center of gravity, mass moments of inertia, products of inertia, principal axes and center of percussion of assemblies of irregularly shaped parts (not necessarily touching nor in the same plane) with respect to arbitrarily selected 3-axis coordinate systems. The solids may either be drawn at the graphics screen, or described and read in on punched cards, or generated by attaching (and modifying) an existing file of parts. The inputted solids may be accumulated with other solids and their combined properties calculated.

This paper describes the techniques used in, and operation of, the PIPS program along with examples of the successful application of PIPS to provide engineering support to the design analysis of problems associated with hardware components and assemblies.

# PROPERTIES OF IRREGULAR PLANAR SOLIDS



## THE PROBLEM

### A. Background

PIPS is an interactive computer graphics program for the calculation of the mass properties of generalized shaped planar solids. The applications of this program at Picatinny support the dynamic evaluations of fuze components and assemblies and missile and shell warhead sections. The need for this program illustrates the fact that even theoretically basic and familiar engineering calculations are at times, not only difficult, but nearly impossible to accurately and expeditiously accomplish in a design room environment. Calculations for mass, CG, and inertial terms are familiar to engineers, designers, and students, but since actual design components perversely do not conform to "nice" uniform shapes, determining these properties of munition parts is a difficult and imposing job. This program offers a considerable savings in development time (and expense) and permits the analyst to "stay on top" of the changing design, thus optimizing final hardware performance.

There are no restrictions to the contours, materials, assemblies of the planar solids, and combinations of complicated convoluted shapes and materials that are routinely handled by the PIPS program.

Any discussion of a computer program should rightfully include a description of the computer system upon which it operates. There are two graphics systems in use at Picatinny Arsenal: a Control Data Corporation 274/1700/6500 refresh graphics facility (fig.2) and a network of Tektronix 4014 storage tube graphics stations (fig.4).

PIPS was originally written for the Control Data Corporation (CDC) facility operating under Scope 3.4.1, IGS Version 2, employing 30 overlays and 46K octal of 60 bit words of storage. The program was later rewritten for the Tektronix Terminal Control System (TCS) package, and required 13 overlays and 56K octal storage. The CDC 274 refresh graphics terminal is a twenty inch diameter screen with a light pen and alphanumeric and function keyboards. It is driven by a mini-computer, a CDC 1700, which is a satellite to the main or host computer, the CDC 6500, which "houses" the problem programs. The problem programs are those programs (stress analysis, circuit design, etc..) to which graphical techniques are being implemented.

The Arsenal's network of Low Cost Graphics Terminals (LCGT) are Tektronix 4014 storage tubes driven directly by the CDC 6000. They feature a nineteen inch diagonal screen, a thumb wheel controlled cross-hair cursor, and an alphanumeric keyboard. Work at the tube is supported by 30"x40" data tablets for digitizing input and "quick look" electrostatic copying devices.

# **PRESENT LARGE SCALE IG SYSTEM**

## **CDC 274/1700/6500 FACILITY**

- **274 DIGIGRAPHICS CRT CONSOLE**
  - **20 INCH DIAMETER SCREEN (REFRESH)**
  - **4096 UNITS OVER 20 INCHES**
  - **LIGHT PEN**
  - **ALPHANUMERIC KEYBOARD**
  - **FUNCTION KEYBOARD**
- **1700 DIGITAL COMPUTER**
  - **SATELLITE**
  - **ITEM IDENTIFICATION HANDLER**
  - **GRAPHIC DRIVERS**
  - **CARD READER, LINE PRINTER, CARD PCH**
- **6500 (HOST) DIGITAL COMPUTER**
  - **PROBLEM PROGRAM**

**FIGURE 1.**

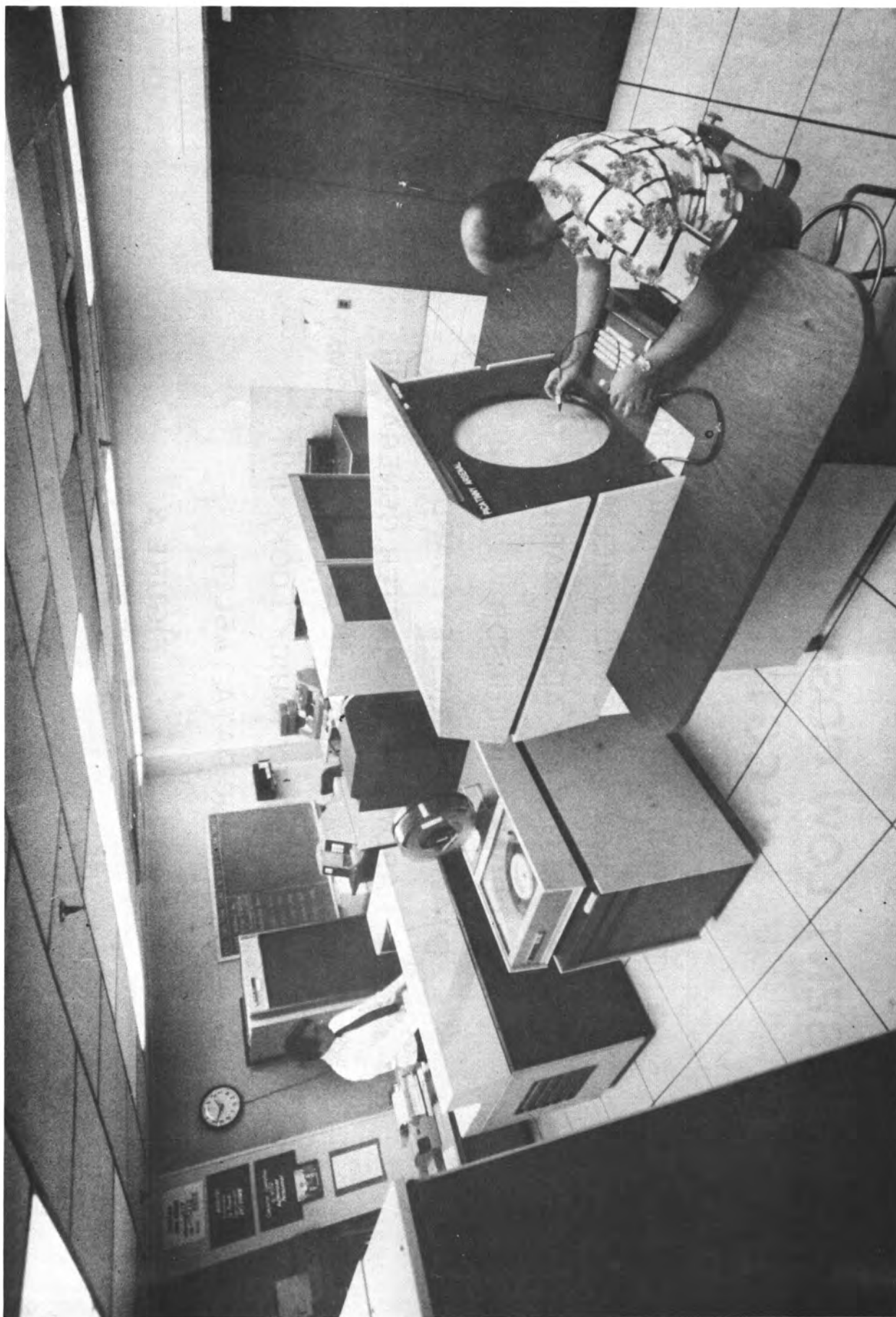


FIGURE 2. LARGE SCALE GRAPHICS FACILITY

# **PRESENT LOW COST GRAPHICS TERMINAL (LCGT) NETWORK**

- TEKTRONIX 4014 CRT TERMINAL CONFIGURATION
  - 19 INCH DIAGONAL SCREEN (STORAGE TUBE)
  - 12 BIT X & Y ADDRESSABILITY (4096)
  - CROSS HAIR CURSOR
  - ALPHANUMERIC KEYBOARD
  - HARDWARE CHARACTER GENERATOR
  - SYNC/COMM INTERFACE (SW SEL COMM)
  - HARD COPY (QUICK LOOK) UNIT
  - GRAPHIC DATA TABLET

**FIGURE 3.**



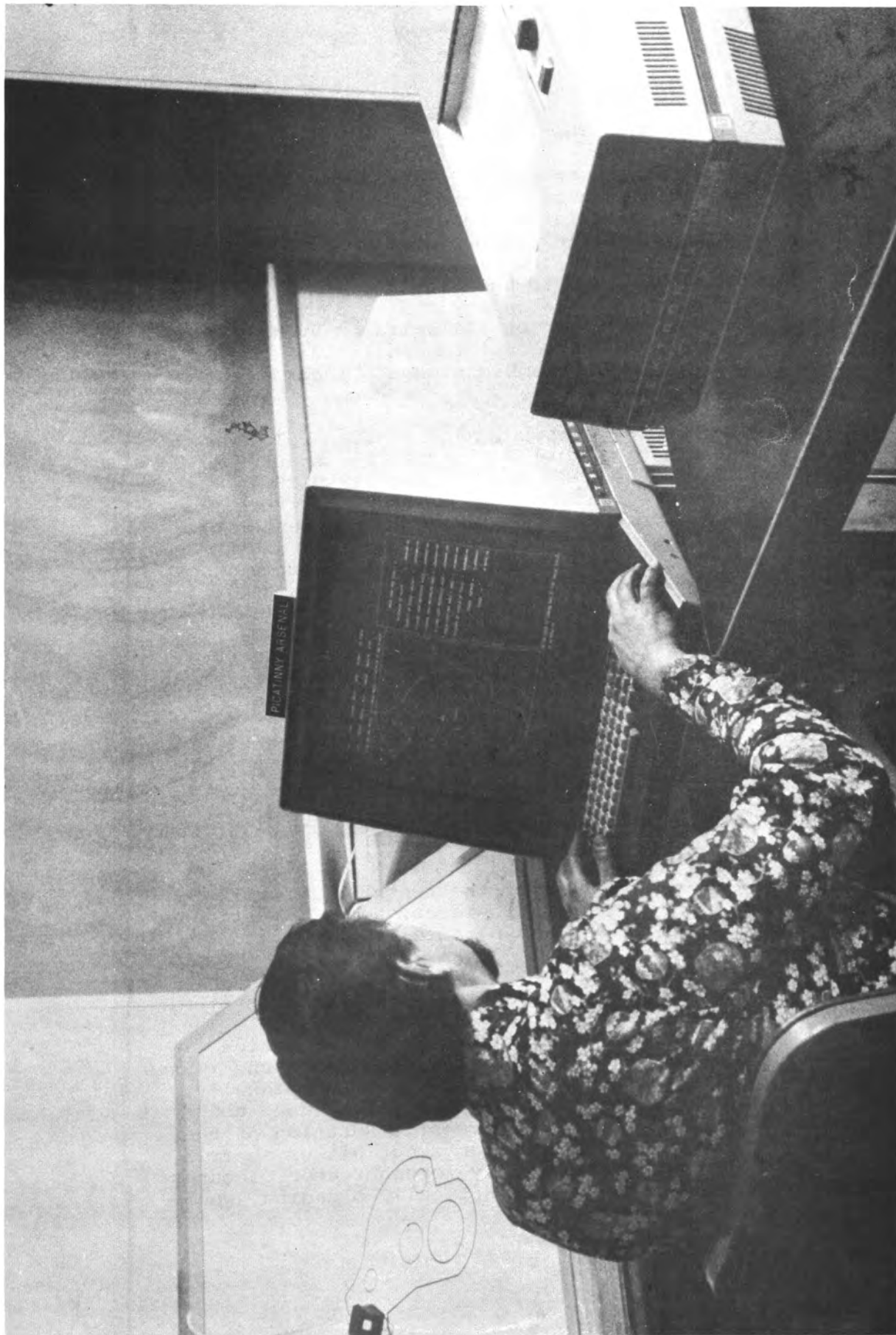


FIGURE 4. LOW COST GRAPHICS STATION

## B. Method of Solution: THE PIPS ALGORITHM

PIPS employs what is termed a "slice method": the solid whose properties are to be determined is sliced by the designer with a series of selectively spaced parallel planes into parallel slices or planar solids (fig.5). These slices are handled by the PIPS code as separate pieces whose properties are to be calculated. The designer may then recombine the pieces in any and as many combinations as desired to either reconstruct the solid or even design a new solid. The mass properties of the individual pieces as well as those of any chosen combination of pieces are calculated by the PIPS program.

Contour, thickness, density, and location relative to the plane of the graphics screen comprise the input data for a piece. All the pieces are described to PIPS in terms of their X,Y, and Z coordinates in the graphic screen's 3 orthogonal axis system, with the origin at the center of the screen. The Z axis is normal to the screen and the X (positive, horizontally to the right) and Y (positive, vertically up) axes lie in the plane of the graphics screen. The pieces are constant thickness slices (the thickness may vary from one piece to the other, as may the density) with generally unrestricted contours, made up of straight lines and circular arc segments in any arrangement. The pieces may have any number of holes (called inner contours) - again of arbitrary shape. Although the pieces may be individually "located" any distance in front of or behind the CRT screen, they must be parallel to the plane of the screen.

All calculations are made with respect to two -3 (orthogonal) axis systems.

One system has its origin at the center of gravity (CG) of the piece, or assembly or pieces, being investigated with the X and Y axes in a plane parallel to the plane of the piece(s).

The designer selects the location of the other system, hence-forth referred to as the User Reference System. The origin of this system may be anywhere (X,Y,Z) in three-dimensional space that the user desires with the X and Y axes also parallel to the plane of the piece(s). Additionally, the User Reference System may be angularly displaced about its Z axis, by any angle (A, in degrees). Rotation of the parts or of the User Reference System about either their X or Y axes would produce artistically foreshortened pictures on the screen which, although aesthetically pleasing, would

# PIPS' SLICE METHOD

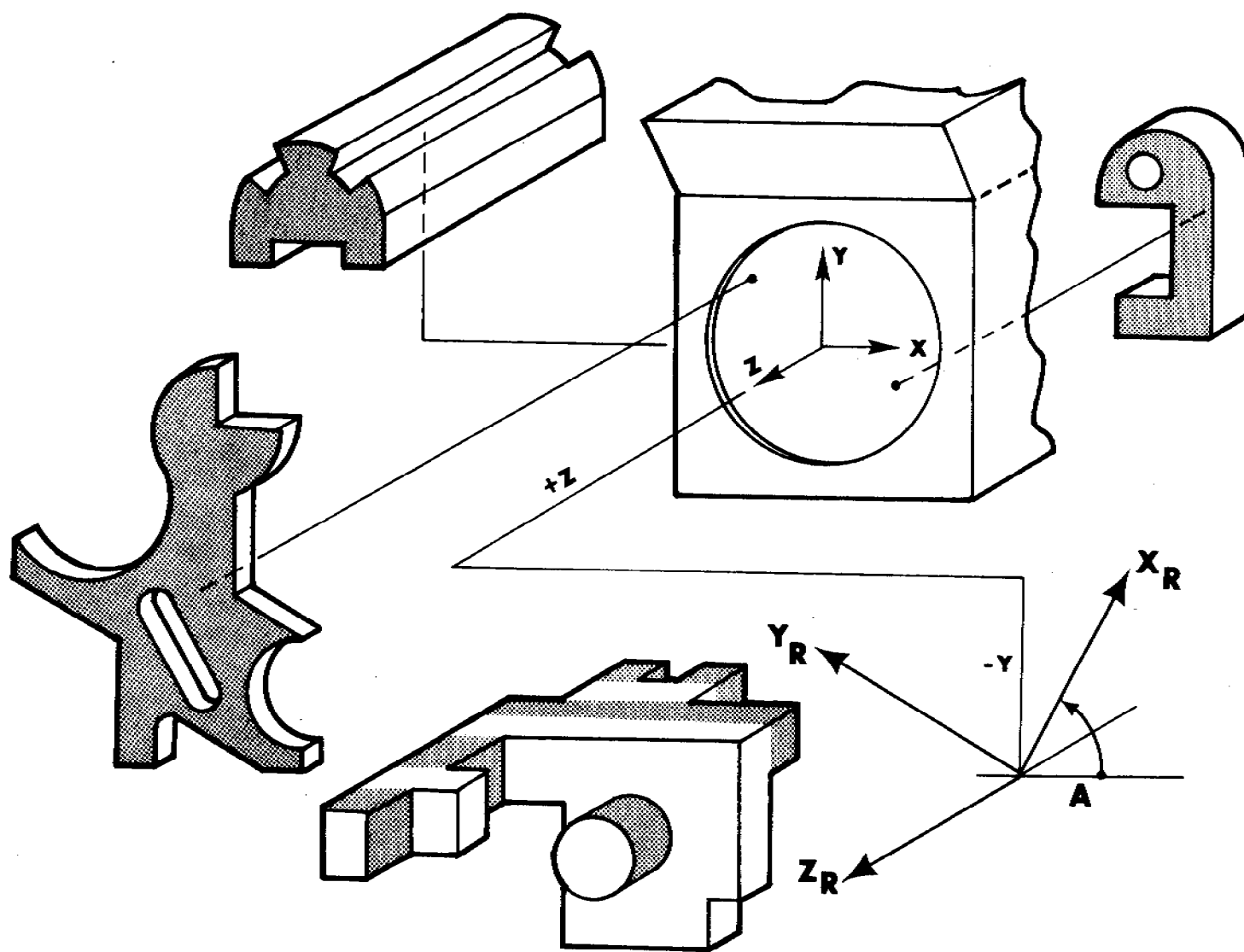


FIGURE 5.

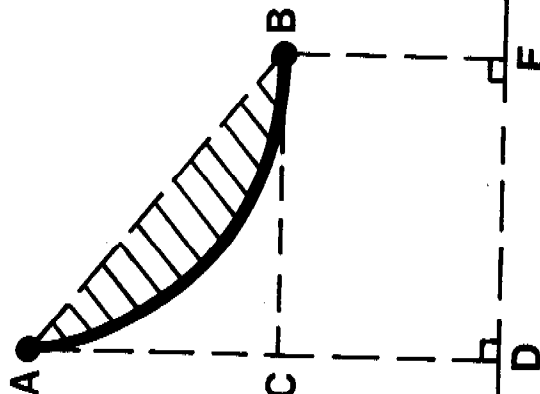
add nothing to engineering understanding of the problem - and add tremendously to programming complexity. Thus, four displacement vectors (three linear and one angular) are permitted for the User Reference System out of a possible six.

It is convenient to initially confine the discussion of the PIPS' algorithm to two-dimensional (plane) polygons and later, by adding thickness to the polygons, extend the explanation to solid bodies.

The outer and inner contours of any shape input to PIPS may be described as contiguous straight line and circular arc segments. The program drops perpendiculars from the end points of the line segments to a base line below. The area enclosed by the line segment, the two perpendiculars, and the base line is automatically partitioned into right triangles, rectangles, and circular segments. The section properties of these well known BASIC SHAPES (fig.6) are in the literature.



All contour lines are developed; that is, drawn from initial point to end point. PIPS establishes the X (horizontal) development of the first line segment as what is called the "ADD" direction. The section properties of the areas under all line segments developed in that direction are added together. The section properties of areas under all line segments developed in the opposite or "SUBTRACT" direction are subtracted from this total. The summation of all this addition and subtraction produces the section properties of the enclosed area of the plane polygon. This is illustrated in the figure entitled SUMMING CONVENTION. (fig.7). To ensure positive values for area and section properties the code uses the absolute values of the summation results. (This over simplified explanation does not, of course, hold for product of inertia calculations. More detailed computer logic is employed to produce the properly signed result). Needless to say, all the bookkeeping of adding, subtracting, re-adding, etc of section properties poses no strain to the user as it is transparently, automatically performed for him by the code.

When thickness of the areas is introduced, the line segments become edges. The areas under the line segments are extrapolated from triangles, rectangles, and circular segments to wedges, rectangular parallelopipeds (boxes), and cylindrical segments. The mass properties of these basic solids are also available in the literature or may be derived, and it is these equations that are coded into the PIPS algorithm (figs. 9-14).

[illegible]

**FIGURE 6.**

**SUMMING CONVENTION**

 ADD  
 SUBTRACT

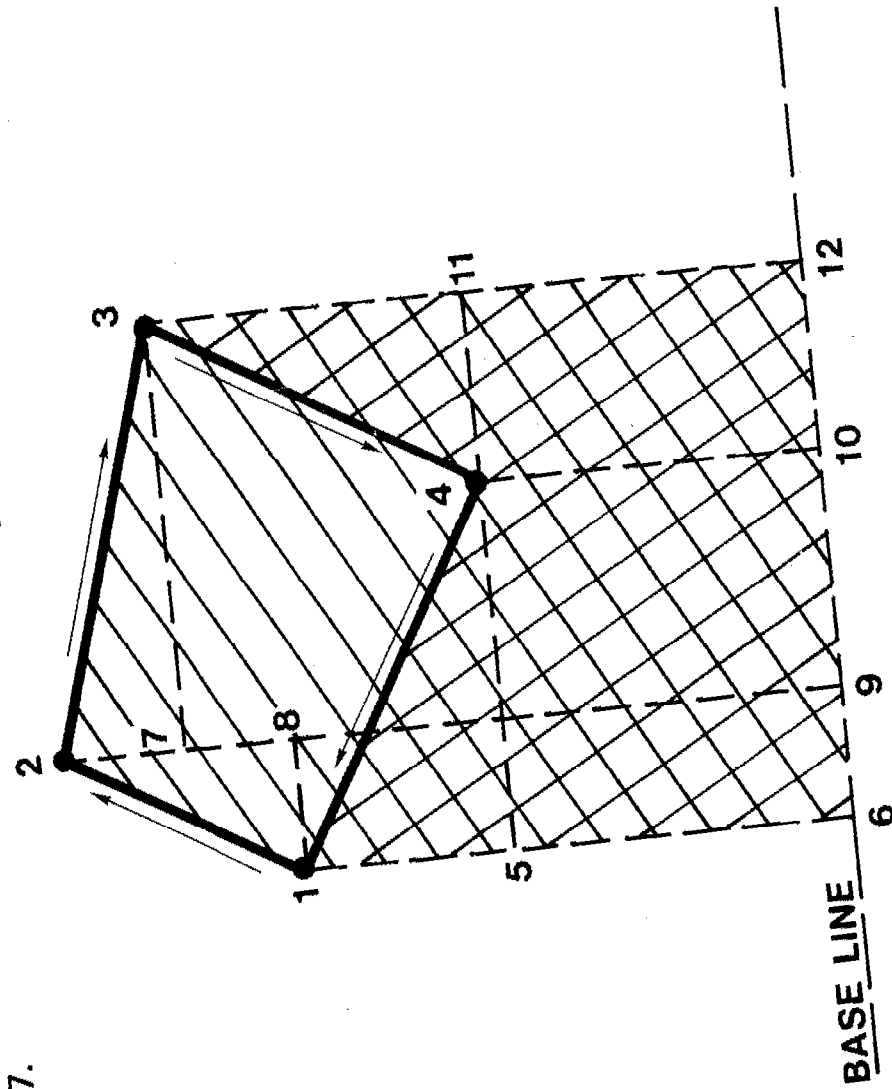


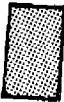


FIGURE 7.

ILLUSTRATIVE SLICE

	ADD		SUBTRACT		RE-ADD
-------------------------------------------------------------------------------------	-----	-----------------------------------------------------------------------------------	----------	-----------------------------------------------------------------------------------	--------

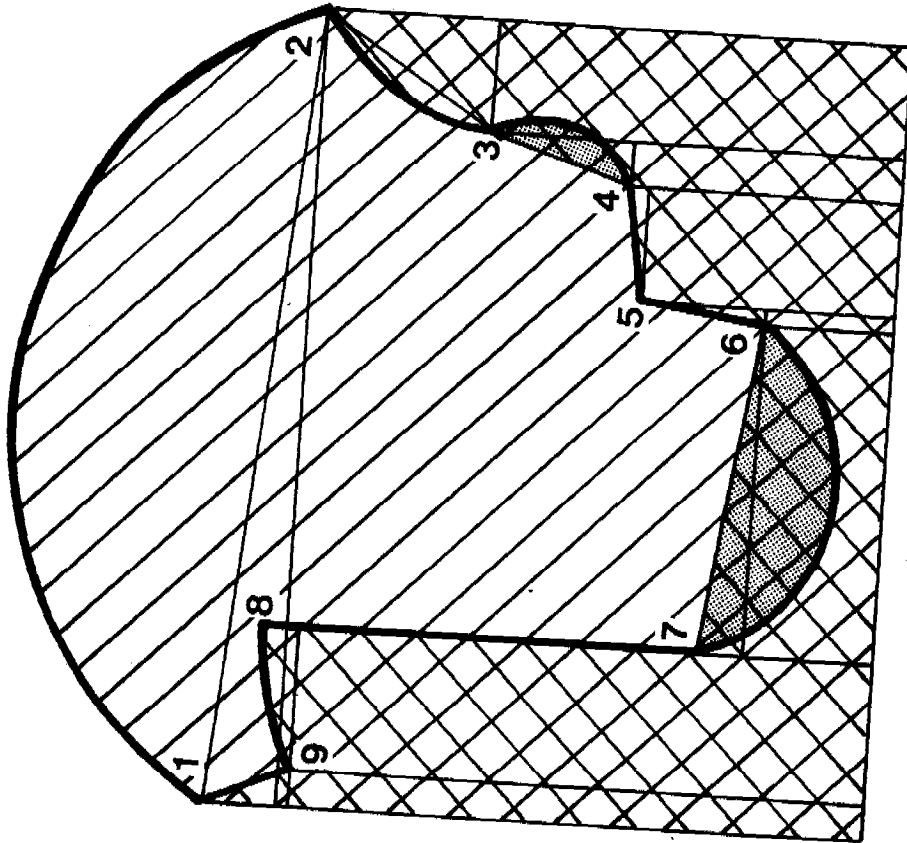


FIGURE 8.

## WEDGE (A)

$\rho$  = MASS DENSITY

$$M = \text{MASS} = \frac{1}{2} \rho x y z$$

C.G.  $\bar{x} = \frac{x}{3}$

$$\bar{y} = \frac{y}{3}$$

$$\bar{z} = \frac{z}{2}$$

MASS MOMENTS OF INERTIA ABOUT AXES THROUGH C.G.

$$I_{xx} = \frac{1}{36} M (y^2 + 2z^2)$$

$$I_{yy} = \frac{1}{36} M (x^2 + 2z^2)$$

$$I_{zz} = \frac{1}{18} M (x^2 + y^2)$$

PRODUCTS OF INERTIA ABOUT PLANES CONTAINING C.G.

$$I_{xy} = -\frac{1}{72} \rho x^2 y^2 z = -\frac{1}{36} M xy$$

$$I_{yz} = 0.0$$

$$I_{xz} = 0.0$$

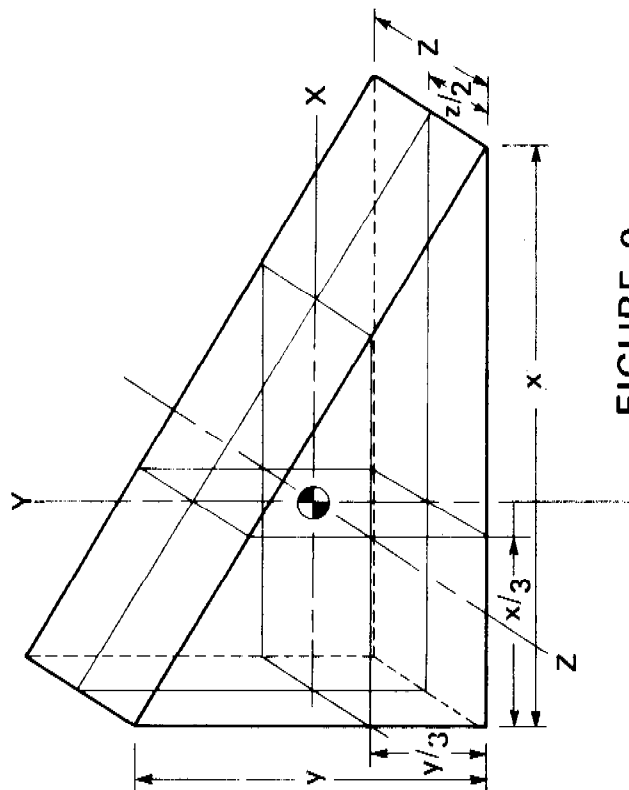


FIGURE 9.



## WEDGE (B)

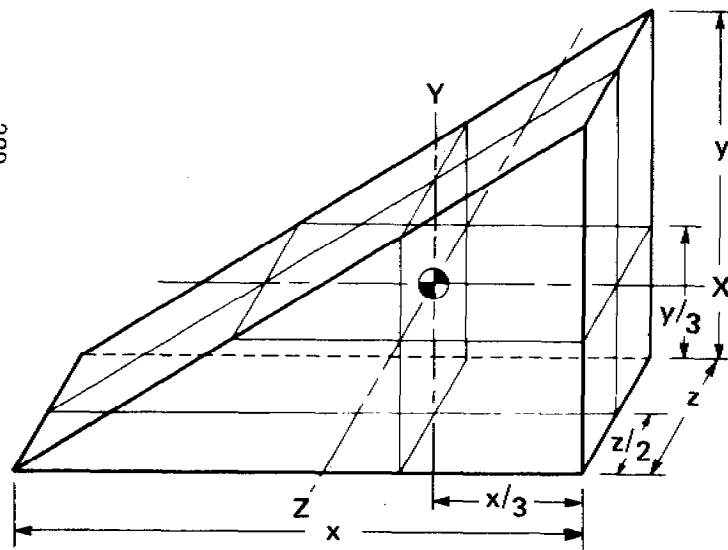
$\rho$  = MASS DENSITY

$$M = \text{MASS} = \frac{1}{2} \rho x y z$$

C.G.  $\bar{x} = \frac{2x}{3}$

$$\bar{y} = \frac{y}{3}$$

$$\bar{z} = \frac{z}{2}$$



MASS MOMENTS OF INERTIA ABOUT AXES THROUGH C.G.

$$I_{xx} = \frac{1}{36} M (y^2 + 2z^2)$$

$$I_{yy} = \frac{1}{36} M (x^2 + 2z^2)$$

$$I_{zz} = \frac{1}{18} M (x^2 + y^2)$$

PRODUCTS OF INERTIA ABOUT PLANES CONTAINING C.G.

$$I_{xy} = \frac{1}{72} \rho x^2 y^2 z = \frac{1}{36} M x y$$

$$I_{yz} = 0.0$$

$$I_{xz} = 0.0$$

FIGURE 10.

# BOX

$\rho$  = MASS DENSITY

$M$  = MASS =  $\rho xyz$

CENTER OF GRAVITY :  $\bar{x} = x/2$

$\bar{y} = y/2$

$\bar{z} = z/2$

MOMENTS OF INERTIA ABOUT AXES THROUGH C.G.

$$I_{xx} = \frac{1}{12} M (z^2 + y^2)$$

$$I_{yy} = \frac{1}{12} M (x^2 + z^2)$$

$$I_{zz} = \frac{1}{12} M (x^2 + y^2)$$

PRODUCTS OF INERTIA WITH RESPECT TO THE PLANES CONTAINING THE C.G.

$$I_{xy} = 0.0$$

$$I_{yz} = 0.0$$

$$I_{xz} = 0.0$$

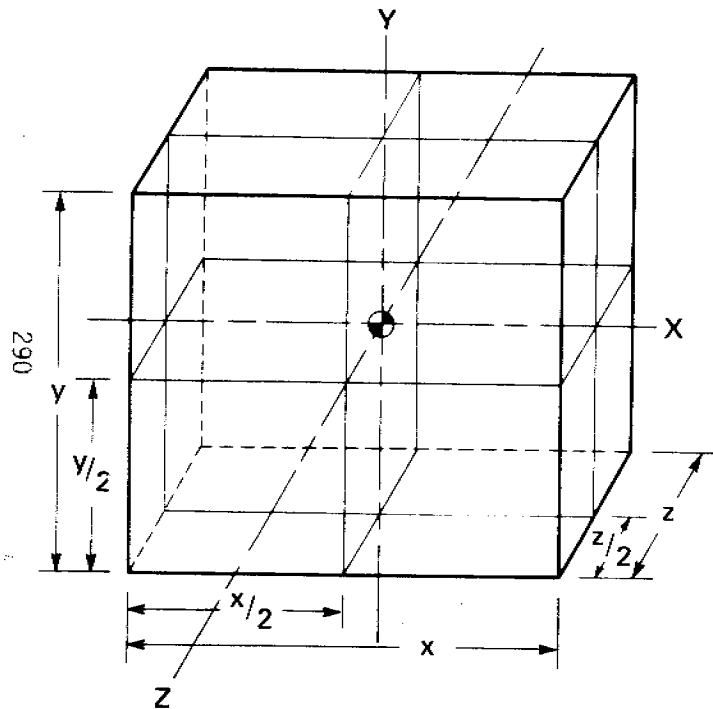
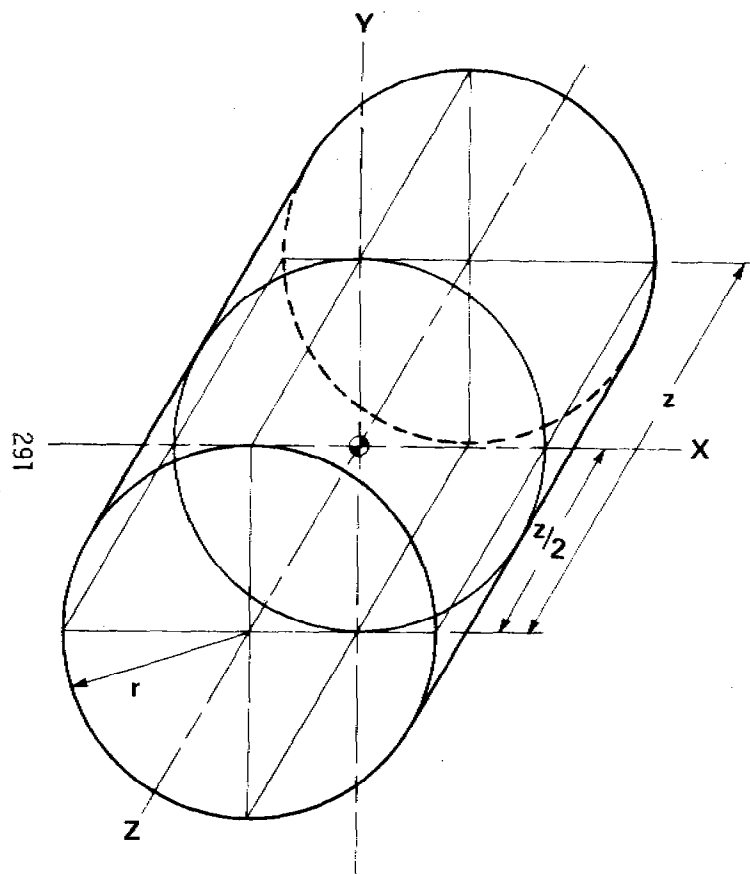


FIGURE 11.



## CYLINDER

$\rho$  = MASS DENSITY

$$M = \text{MASS} = \rho \pi r^2 z$$

C.G.  $\bar{x}$  = CENTER OF CIRCLE

$\bar{y}$  = CENTER OF CIRCLE

$$\bar{z} = z/2$$

MOMENTS OF INERTIA ABOUT AXES THROUGH C.G.

$$I_{xx} = \frac{1}{12} M (3 r^2 + z^2)$$

$$I_{yy} = \frac{1}{12} M (3 r^2 + z^2)$$

$$I_{zz} = \frac{1}{12} M (6 r^2)$$

PRODUCTS OF INERTIA ABOUT PLANES CONTAINING C.G.

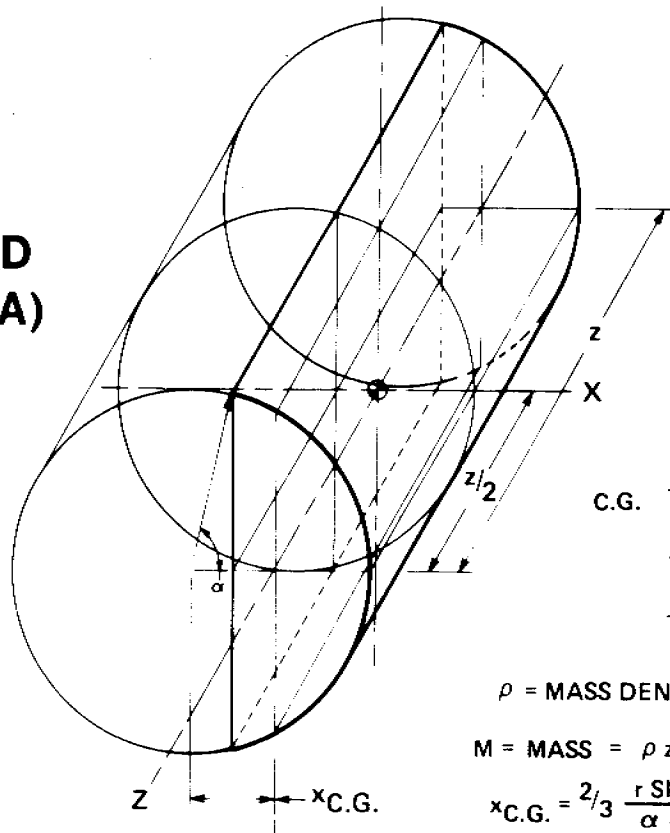
$$I_{xy} = 0.0$$

$$I_{yz} = 0.0$$

$$I_{xz} = 0.0$$

FIGURE 12.

## SEGMENTED CYLINDER (A)



$$\text{C.G. } \bar{x} = x_{\text{C.G.}}$$

$$\bar{y} = 0.$$

$$\bar{z} = z/2$$

$\rho$  = MASS DENSITY

$$M = \text{MASS} = \rho z r^2 (\alpha - \sin \alpha \cos \alpha)$$

$$x_{\text{C.G.}} = \frac{2}{3} \frac{r \sin^3 \alpha}{\alpha - \sin \alpha \cos \alpha}$$

FIGURE 13.

MASS MOMENTS OF THE INERTIA ABOUT AXES THROUGH C.G. OF THE CYLINDER  
CONTAINING THE SEGMENT

$$I_{xx} = \frac{\rho r^2 z^3}{12} (\alpha - \cos \alpha \sin \alpha) + \frac{\rho r^4 z}{4} (\alpha - \sin \alpha \cos \alpha - \frac{2}{3} \sin^3 \alpha \cos \alpha)$$

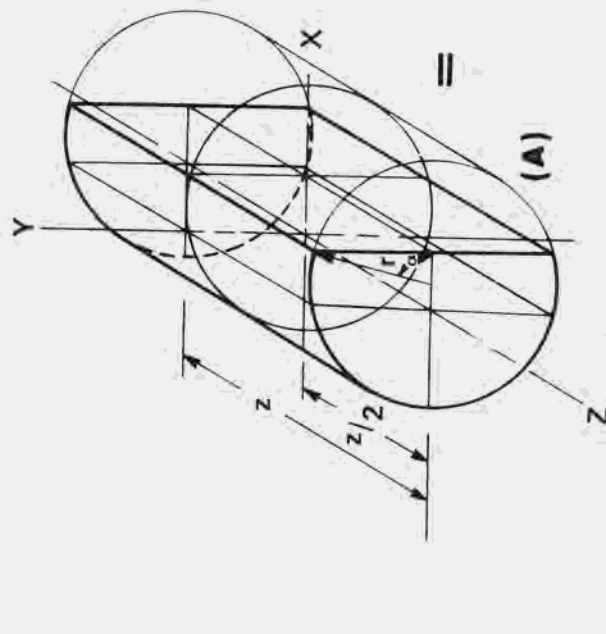
$$I_{yy} = \frac{\rho r^2 z^3}{12} (\alpha - \cos \alpha \sin \alpha) + \frac{\rho r^4 z}{4} (\alpha - \sin \alpha \cos \alpha + 2 \sin^3 \alpha \cos \alpha)$$

$$I_{zz} = \frac{\rho r^4 z}{2} (\alpha - \sin \alpha \cos \alpha + \frac{2}{3} \sin^3 \alpha \cos \alpha)$$

PRODUCTS OF INERTIA ABOUT PLANES THROUGH C.G. OF THE CYLINDER  
THAT CONTAINS THE SEGMENT

$$I_{xy} = I_{xz} = I_{yz} = 0.0$$

# SEGMENTED CYLINDER (B)



$\rho$  = MASS DENSITY

$$\text{MASS} = \rho \pi r^2 z (\pi - \alpha + \sin \alpha \cos \alpha)$$

MASS MOMENTS OF INERTIA OF (A) ABOUT C.G. OF THE CYLINDER IT IS CONTAINED IN

$$I_{xx} (A) = I_{xx} (B) - I_{xx} (C)$$

$$I_{yy} (A) = I_{yy} (B) - I_{yy} (C)$$

$$I_{zz} (A) = I_{zz} (B) - I_{zz} (C)$$

$$I_{xy} (A) = I_{xz} (A) = I_{yz} (A) = 0.0$$

ALL OF WHICH ARE ABOUT THE C.G. OF THE CYLINDERS CONTAINING (A), (B), (C)

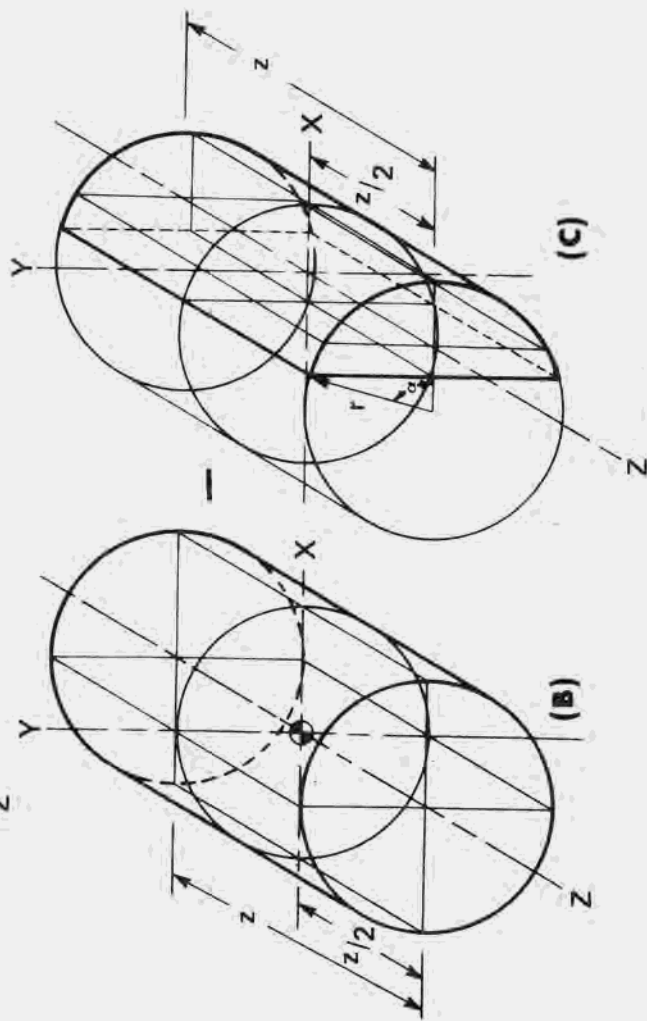


FIGURE 14.

### C. Step-By-Step Illustrative Example

Consider the rotor sub-assembly of the M557 Point Detonating (PD) Fuze as a typically convoluted piece of fuze hardware. A dynamic analysis would first require the determination of its more prosaic mass properties. The best way to appreciate the strength of the PIPS code is to "walk through" the processing of a sample slice of the sub-assembly. The following sequence of photos provide a glimpse of such an analysis performed on the CDC 274 refresh graphics screen. The prints of the close-ups of the screen are negative prints (black line on white background) for ease of publication.

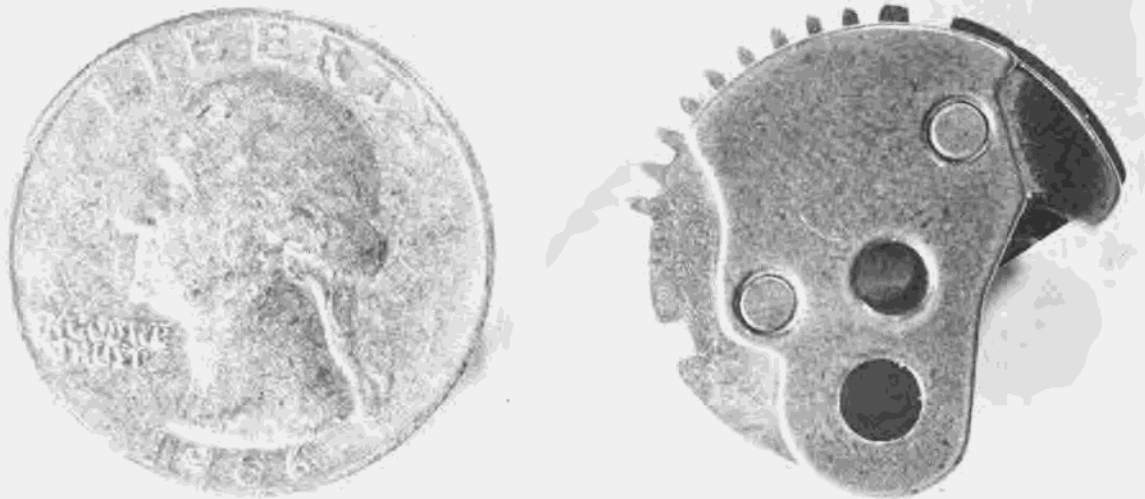


FIGURE 15. ILLUSTRATIVE EXAMPLE

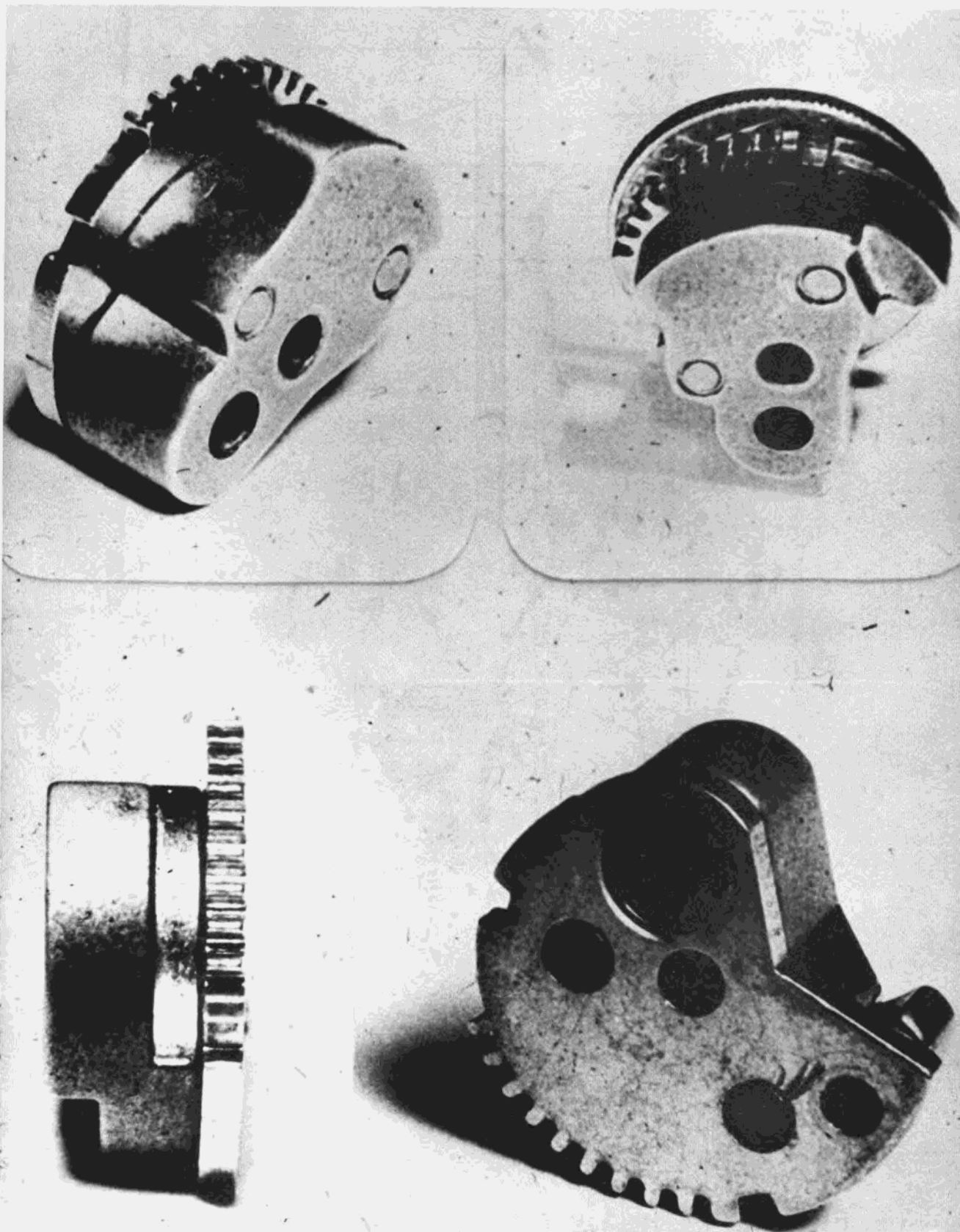
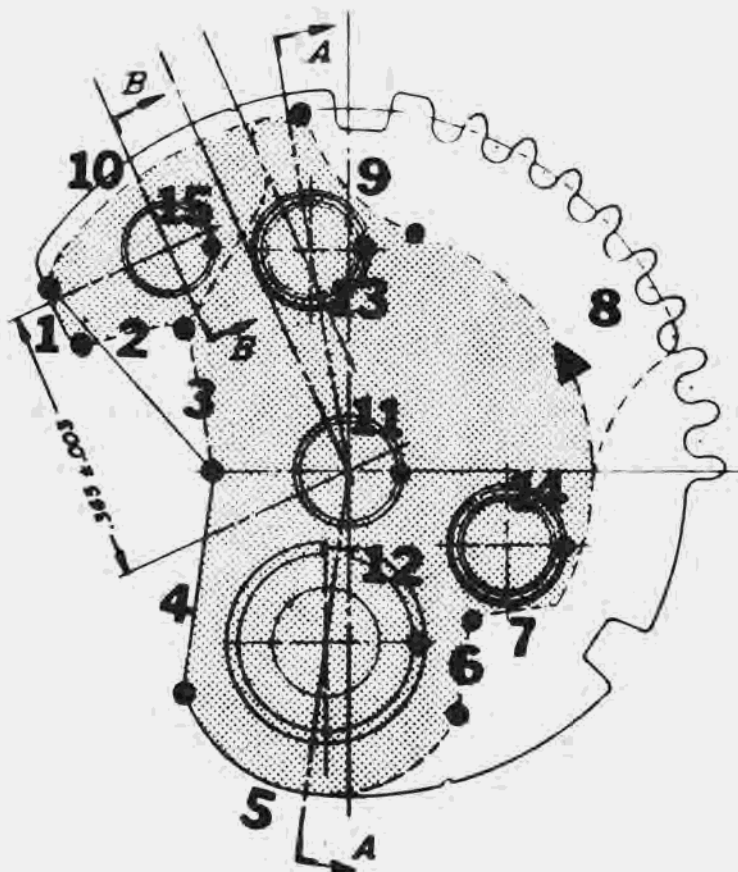
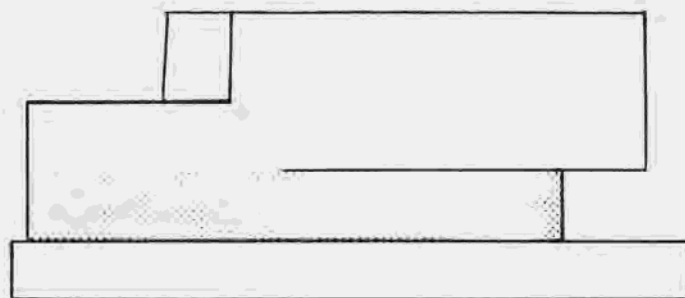


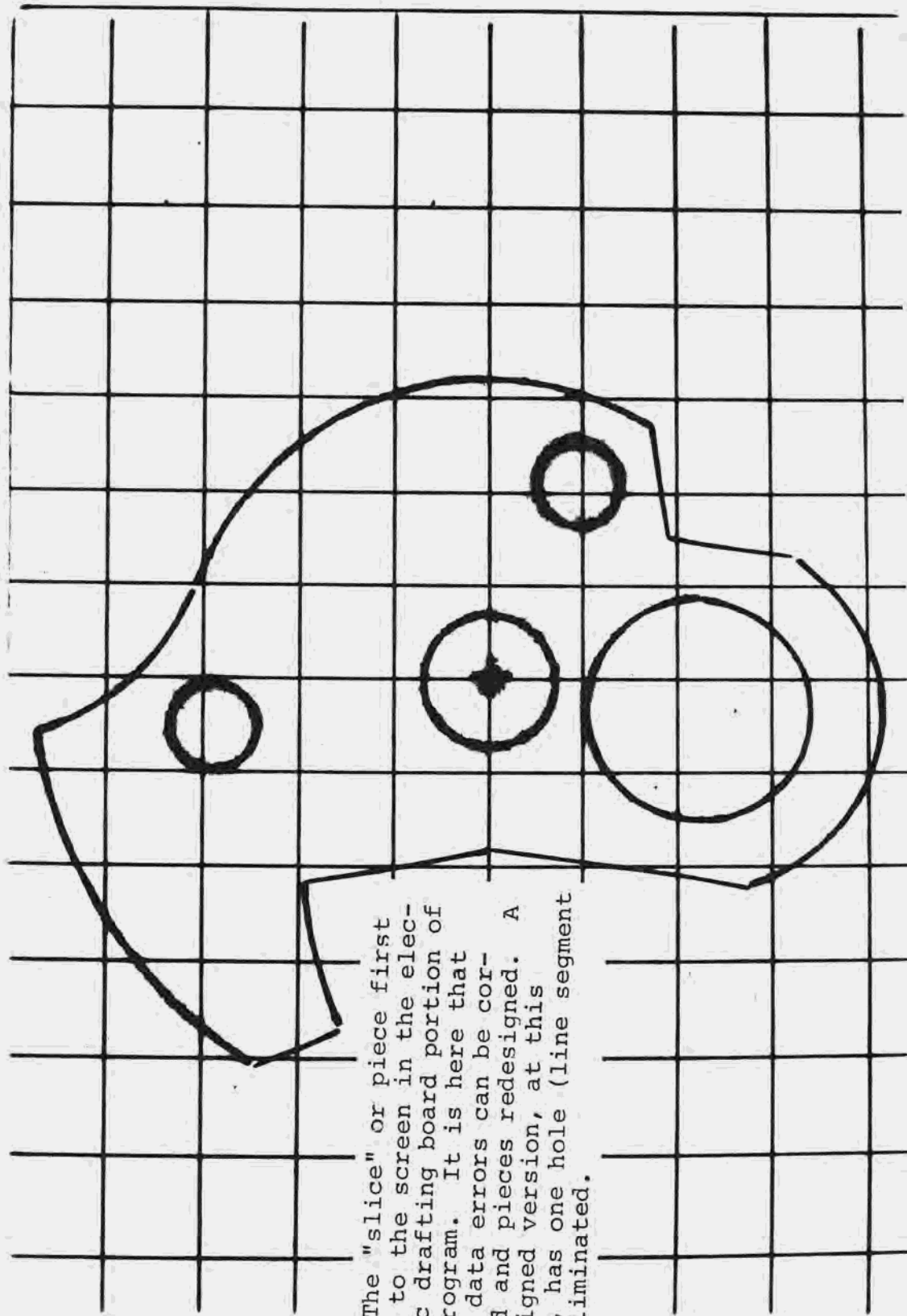
FIGURE 16. ROTOR SUB-ASSEMBLY



Only fifteen (15) straight line and circular arc segments are required to completely describe the outer and inner contours of one slice (shaded portion) of the rotor sub-assembly.

FIGURE 17. ILLUSTRATIVE SLICE





The "slice" or piece first comes to the screen in the electronic drafting board portion of the program. It is here that input data errors can be corrected and pieces redesigned. A redesigned version, at this point, has one hole (line segment 15) eliminated.

FIGURE 18. SLICE ON ELECTRIC DRAFTING BOARD

Control is transferred to the basic PIPS routines. The mass properties are determined and the results displayed below the picture of the piece.

PICATINNY ARSENAL

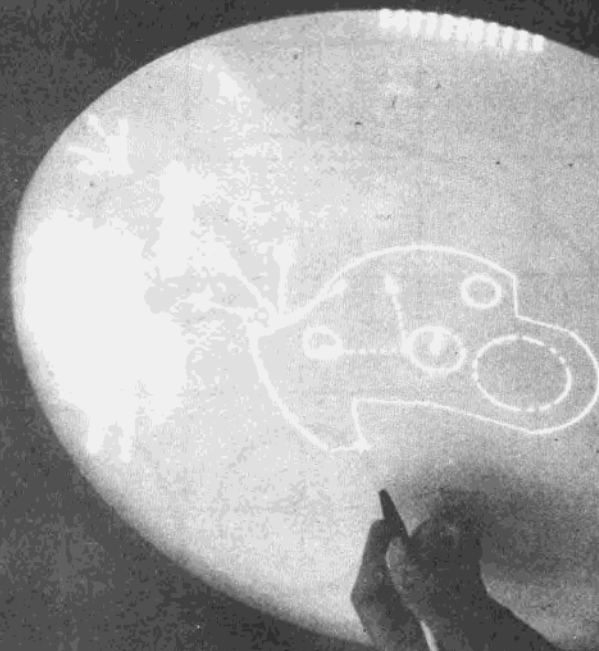
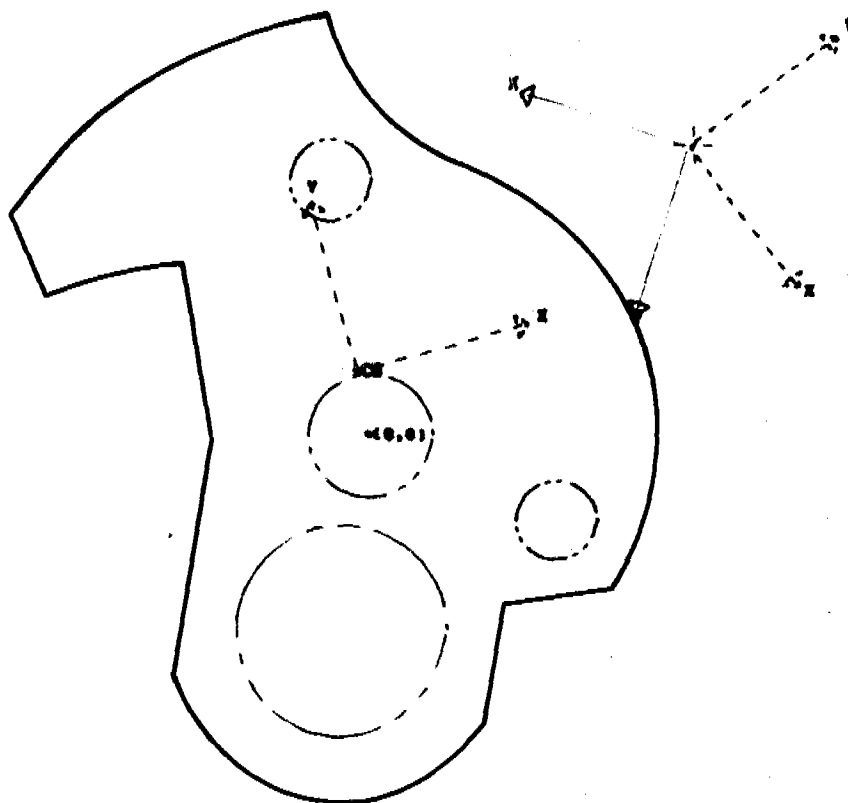


FIGURE 19. PIPS IN USE

OF REFERENCE AXIS

PACKING CROSS AS COORDINATES  
OF A PT. ON THE X-AXIS



The solid arrows are the X and Y axes of the User Reference System. The two pair of dashed vectors are the principal axes drawn through the CG of the piece and through the User Reference Origin.

PRINCIPAL AXIS ANGLES  
AT REF. = -53.58771  
AT C.G. = 16.68534  
GEOMETRY  
AREA X,Y = 29510  
VOLUME = 29510E-01  
MASS = 17682E-04  
WEIGHT = 67872E-02  
CGX = -14440E-01  
CGY = 72180E-01  
CGZ = -50000

REFERENCE AXES  
IXX = .298074E-06  
IYY = .298012E-06  
IZZ = .492954E-06  
IXZ = 0.  
IYZ = 0.  
IXY = .158052E-06  
IX/XZ = .299239E-06  
IY/YZ = .196657E-06  
IX/YZ = .155725E-06

AXES THROUGH C.G.  
IXX = .884710E-06  
IYY = .601444E-06  
IZZ = .132737E-06  
IXZ = 0.  
IYZ = 0.  
IXY = -115902E-06  
IX/XZ = 7.61692E-06  
IY/YZ = .994983E-06  
IX/YZ = -1.93808E-06

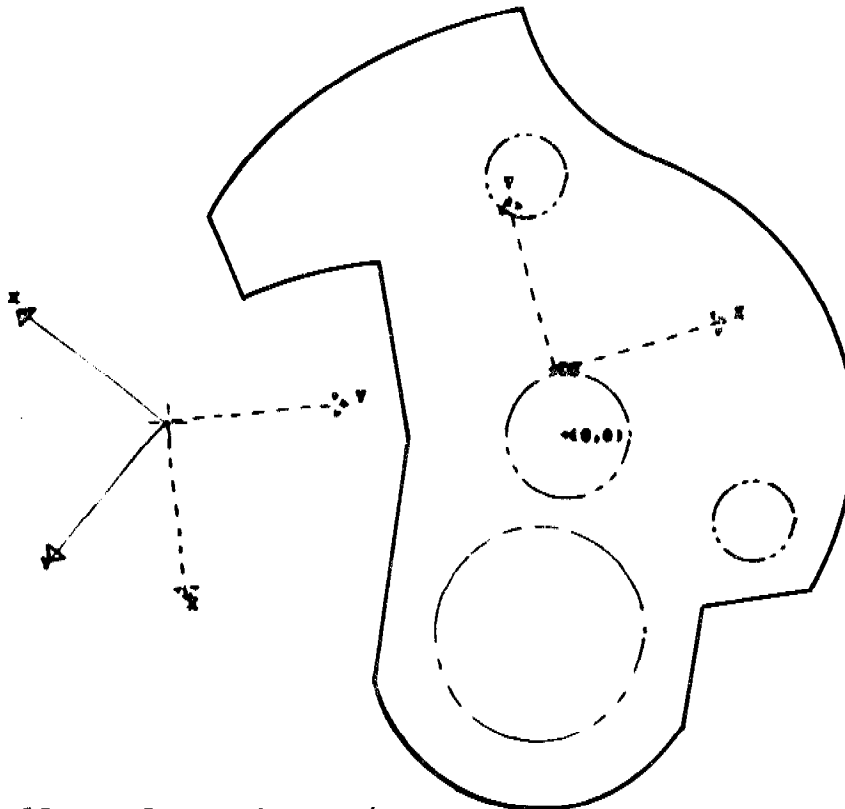
PRINCIPAL AXES  
OF REFERENCE AXES  
IXX = .411704E-06  
IYY = .841008E-06  
IYZ = .169407E-20  
OF C.G. AXES  
IXX = .899304E-06  
IYY = .467292E-06  
IXY = .127658E-20

OS/

FIGURE 20. CLOSE-UP OF SINGLE SLICE RESULTS

TRACKING CROSS AS COORDINATES  
OF REFERENCE AXIS

TRACKING CROSS AS COORDINATES  
OF A PT. ON THE X-AXIS



The effect of any change (con-  
tour, material, thickness, etc.)  
to the piece may be immediately  
seen. Here the User Reference  
Origin is "attached" with the  
light pen and towed (translated)  
to another location. All prop-  
erties are recalculated.

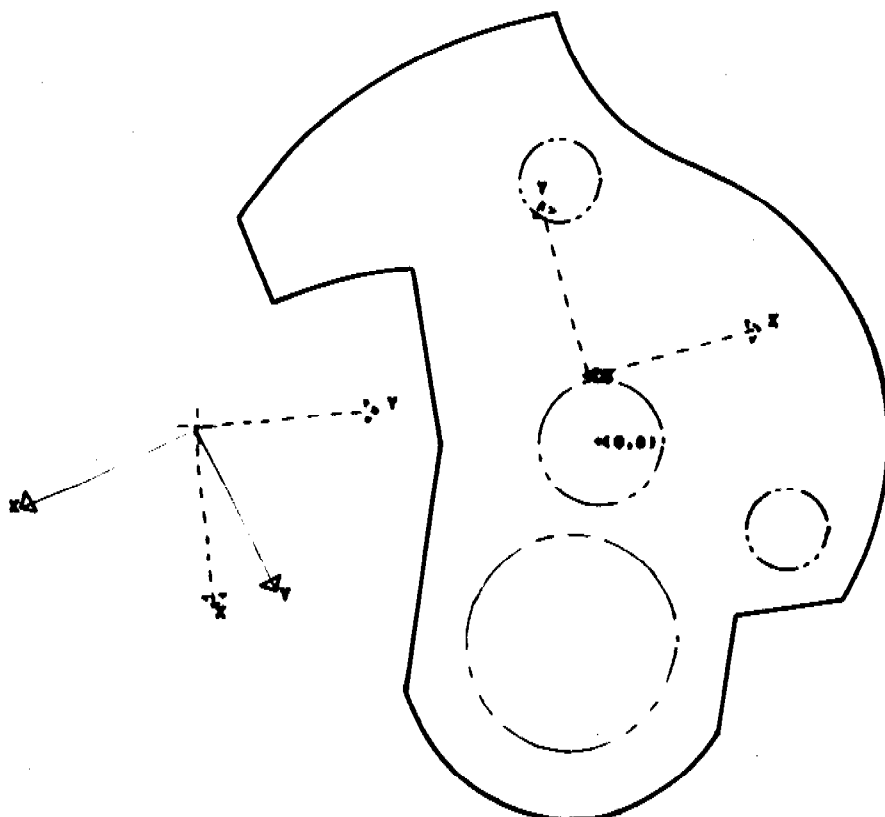
PRINCIPAL AXIS ANGLES	REFERENCE AXES	AXES THROUGH C.G.	PRINCIPAL AXES OF REFERENCE AXES
AT REF. = -83.14641	IXX = .791098E-06	IXX = .854718E-06	IXX = .399578E-06
AT C.G. = 16.66834	IYY = .385877E-05	IYY = .881966E-06	IYY = .878431E-06
GEOMETRY	IYY = .474506E-05	IYY = .132737E-05	IYY = -.608300E-06
AREA X,Y = .29510	IXZ = 0	IXZ = 0	IXZ = 0
VOLUME = .29510E-01	IYZ = 0	IYZ = 0	IYZ = 0
MASS = .17582E-04	IXY = .357620E-06	IXY = -.118993E-06	IXY = .899384E-06
WEIGHT = .67872E-02	IX/X = .240373E-05	IX/X = .698361E-06	IXY = .447292E-06
CGX = .14440E-01	IY/Y = .227064E-05	IY/Y = .751219E-06	IYV = 0
CGY = .79180E-01	IX/Y = .154047E-05	IX/Y = -.198827E-06	
CGZ = .50000			

08/

FIGURE 21. TRANSLATION OF USER REFERENCE ORIGIN

TRACKING CROSS AS COORDINATES  
OF REFERENCE AXIS

TRACKING CROSS AS COORDINATES  
OF A PT. ON THE X-AXIS



Here, the User Reference System is rotated about the Z axis to a new orientation. The coordinate system can be relocated (and reorientated) with the light pen or via the keyboard. Again, the displayed results are newly calculated.

PRINCIPAL AXIS ANGLES  
AT REF. = -83.14541  
AT C.G. = 15.48534  
GEOMETRY  
AREA(X,Y) = .89510  
VOLUME = .39510E-01  
MASS = .17500E-04  
WEIGHT = .67872E-02  
CGX = -.14440E-01  
CGY = -.73180E-01  
CGZ = -.50000

REFERENCE AXES  
IXX = .92199E-06  
IYY = .38827E-06  
IZZ = .47484E-06  
IXZ = 0.  
IYZ = 0.  
IXY = .35789E-06  
IX/Y = .12884E-06  
IY/Y = .28487E-06  
IX/Y = -.96232E-06

AXES THROUGH C.G.  
IXX = .88471E-06  
IYY = .88176E-06  
IZZ = .12873E-06  
IXZ = 0.  
IYZ = 0.  
IXY = -.11090E-06  
IX/Y = .87661E-06  
IY/Y = .48806E-06  
IX/Y = .72301E-07

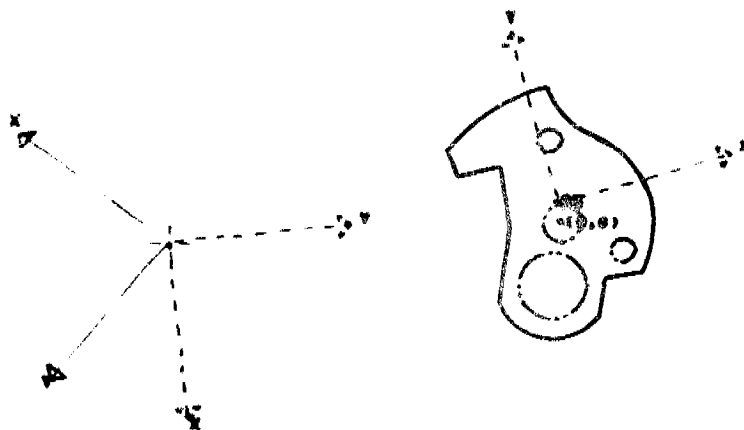
PRINCIPAL AXES  
OF REFERENCE AXES  
IXX = .39957E-05  
IYY = .87662E-06  
IXY = -.50022E-06  
OF C.G. AXES  
IXX = .88730E-06  
IYY = .46729E-06  
IXY = 0.

OS/

FIGURE 22. ROTATION OF USER REFERENCE SYSTEM

OF REFERENCE AXIS

COORDINATES  
OF A PT. ON THE X AXIS



The User Reference System can even be repositioned off the screen - out of the building, if desired. PIPS automatically re-scales the picture to fit within the boundaries of the CRT.

PICK THIS TO GO BACK TO ORIGINAL

<b>PRINCIPAL AXIS ANGLES</b> AT REF. = -83.16757 AT C.G. = 16.65634 <b>GEOMETRIC</b> AREA X,Y: 29610 VOLUME = .29510E-01 MASS = .17582E-04 HEIGHT = .67072E-03 CGX = -.11440E-01 CGY = .72100E-01 CGZ = -.50000	<b>REFERENCE AXES</b> IXX = -131047E-05 IYY = -316100E-04 IZZ = -322990E-04 IXY = 0. IYZ = 0. IZX = 0. IYX = -166234E-05 IXY = -166234E-04 IYZ = -166234E-04 IZY = -166234E-04	<b>AXES THROUGH C.G.</b> Ixx = -864710E-06 Iyy = -581940E-06 Izz = -132737E-06 Ixy = 0. Iyz = 0. Izx = 0. Iyx = -115902E-06 Ixy = -115902E-06 Iyz = 751315E-06 Izy = -115902E-06	<b>PRINCIPAL AXES</b> OF REFERENCE AXES Ixx = 177513E-04 Iyy = 47220E-06 Izz = 9.0171E-04 Ixy = 0. Iyz = 0. Izx = 0. Iyx = 0. Ixy = 0. Iyz = 0. Izy = 0.
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

FIGURE 23. AUTOMATIC SCALING




FIGURE 24. COMBINATION OF SLICES

After analysis and redesign of a piece or slice is completed, it may be stored away in a library of such slices. The designer can "pick" with the light pen any desired combination of slices from the displayed pages of the library. The reconstructed or newly designed assembly is drawn on the screen with its (calculated) mass properties displayed below.

FIGURE 25. FRONT VIEW OF ASSEMBLY

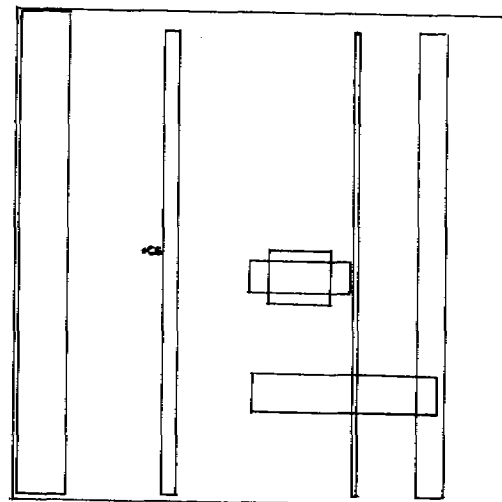
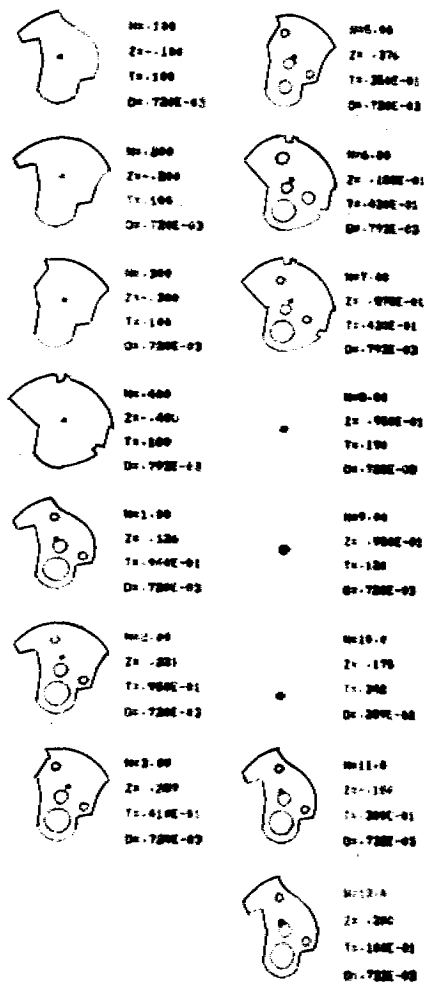
[illegible]





20-5-80  
 21-12-80  
 12-09-81  
 08-12-81

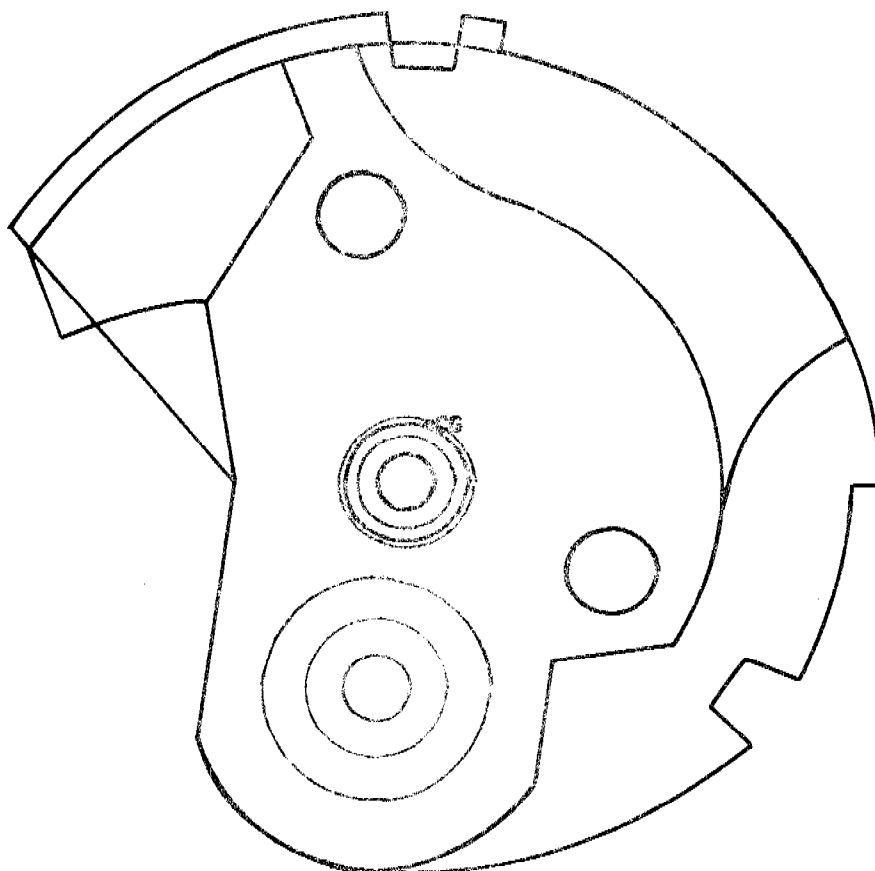
PLST	RETTART	POPE	RE TLOAN	FROM
CHANGE REFERENCE		CHANGE REFERENCE		



		REFERENCE A-12		A-12 THROUGH C-6	
RAIS	0	121	1194-42-01	121	1194-42-01
CSF	0	122	1194-42-02	122	1194-42-02
CSF	0	123	1194-42-03	123	1194-42-03
CSF	0	124	1194-42-04	124	1194-42-04
CSF	0	125	1194-42-05	125	1194-42-05
CSF	0	126	1194-42-06	126	1194-42-06
CSF	0	127	1194-42-07	127	1194-42-07
CSF	0	128	1194-42-08	128	1194-42-08
CSF	0	129	1194-42-09	129	1194-42-09
CSF	0	130	1194-42-10	130	1194-42-10
CSF	0	131	1194-42-11	131	1194-42-11
CSF	0	132	1194-42-12	132	1194-42-12
CSF	0	133	1194-42-13	133	1194-42-13
CSF	0	134	1194-42-14	134	1194-42-14
CSF	0	135	1194-42-15	135	1194-42-15
CSF	0	136	1194-42-16	136	1194-42-16
CSF	0	137	1194-42-17	137	1194-42-17
CSF	0	138	1194-42-18	138	1194-42-18
CSF	0	139	1194-42-19	139	1194-42-19
CSF	0	140	1194-42-20	140	1194-42-20
CSF	0	141	1194-42-21	141	1194-42-21
CSF	0	142	1194-42-22	142	1194-42-22
CSF	0	143	1194-42-23	143	1194-42-23
CSF	0	144	1194-42-24	144	1194-42-24
CSF	0	145	1194-42-25	145	1194-42-25
CSF	0	146	1194-42-26	146	1194-42-26
CSF	0	147	1194-42-27	147	1194-42-27
CSF	0	148	1194-42-28	148	1194-42-28
CSF	0	149	1194-42-29	149	1194-42-29
CSF	0	150	1194-42-30	150	1194-42-30
CSF	0	151	1194-42-31	151	1194-42-31
CSF	0	152	1194-42-32	152	1194-42-32
CSF	0	153	1194-42-33	153	1194-42-33
CSF	0	154	1194-42-34	154	1194-42-34
CSF	0	155	1194-42-35	155	1194-42-35
CSF	0	156	1194-42-36	156	1194-42-36
CSF	0	157	1194-42-37	157	1194-42-37
CSF	0	158	1194-42-38	158	1194-42-38
CSF	0	159	1194-42-39	159	1194-42-39
CSF	0	160	1194-42-40	160	1194-42-40
CSF	0	161	1194-42-41	161	1194-42-41
CSF	0	162	1194-42-42	162	1194-42-42
CSF	0	163	1194-42-43	163	1194-42-43
CSF	0	164	1194-42-44	164	1194-42-44
CSF	0	165	1194-42-45	165	1194-42-45
CSF	0	166	1194-42-46	166	1194-42-46
CSF	0	167	1194-42-47	167	1194-42-47
CSF	0	168	1194-42-48	168	1194-42-48
CSF	0	169	1194-42-49	169	1194-42-49
CSF	0	170	1194-42-50	170	1194-42-50
CSF	0	171	1194-42-51	171	1194-42-51
CSF	0	172	1194-42-52	172	1194-42-52
CSF	0	173	1194-42-53	173	1194-42-53
CSF	0	174	1194-42-54	174	1194-42-54
CSF	0	175	1194-42-55	175	1194-42-55
CSF	0	176	1194-42-56	176	1194-42-56

The slices are not necessarily in the same plane.

**FIGURE 26. SIDE VIEW OF ASSEMBLY**



			REFERENCE AXES		AXES THROUGH C.G.			
MASS	=	.82553E-04	Ixx	=	.677283E-05	Ixx	=	.565217E-05
CGX	=	.25513E-01	Iyy	=	.665654E-05	Iyy	=	.581112E-05
CGY	=	.63638E-01	Izz	=	.733010E-05	Izz	=	.693880E-05
CGZ	=	.97663E-01	Ixz	=	.513710E-06	Ixz	=	.299954E-06
RX	=	0.	Iyz	=	.773133E-06	Iyz	=	.260866E-06
RY	=	0.	Ixy	=	-.393192E-06	Ixy	=	-.532259E-06
RZ	=	0.						

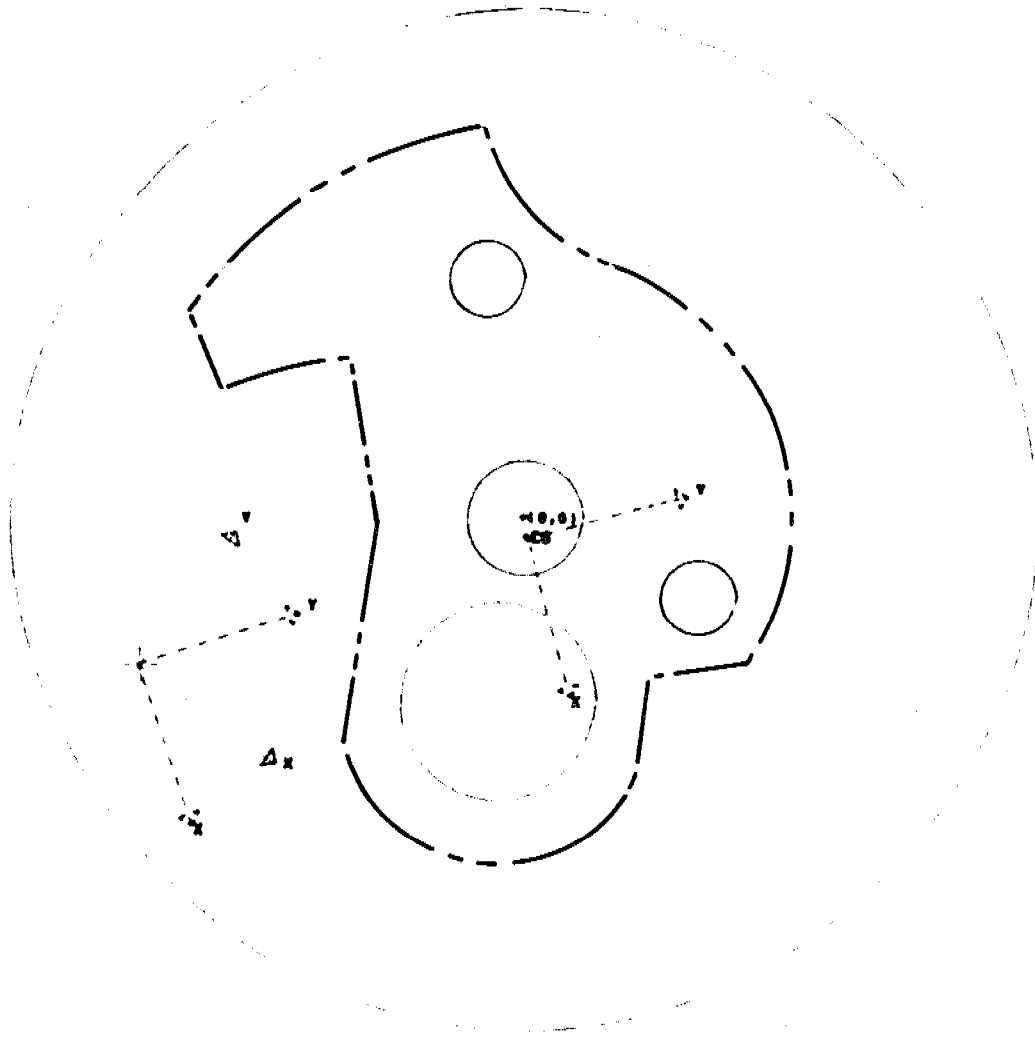
FIGURE 27. CLOSE-UP OF FRONT VIEW  
ASSEMBLY DRAWING

It is possible at any time to return to the electronic drafting board portion of PIPS to redesign a piece or create an entirely new item. As an example, the contour of the recently examined piece is used as the inner contour of an irregular hole in a newly designed circular disk. The four holes in the original piece are now small circular disks of different diameters located within the irregularly shaped inner contour. The new configuration is now a cluster of five solids with all of the PIPS graphics and calculation capabilities (change of contour, material and reference axes, automatic scaling, etc..) available - a cluster can be handled as if it were a single entity.

Finally, a file of any display of a piece or assembly of pieces may be catalogued for use by a plotter program to produce a hard copy of what appears on the screen. (Slide and Poloroid cameras also provide reproductions of the screen contents but their proper utilization require additional skills).

TRACKING CROSS AS COORDINATES  
OF REFERENCE AXIS

TRACKING CROSS AS COORDINATES  
OF A PT. ON THE X-AXIS



PRINCIPAL AXIS ANGLES  
AT REF. = -71.44698  
AT C.G. = -74.64136  
GEOMETRY  
AREA(X,Y) = .07209  
VOLUME = .48064E-01  
MASS = .35235E-04  
WEIGHT = .13602E-01  
CGX = .48762E-02  
CGY = -.24712E-01  
CGZ = 0.

REFERENCE AXES  
IXX = .447407E-05  
IYY = .121209E-04  
IZZ = .163703E-04  
IXZ = -.164054E-05  
IYZ = -.587720E-06  
IXY = .278473E-05  
IX/X/ = .986910E-05  
IY/Y/ = .723167E-05  
IX/Y/ = .428495E-05

AXES THROUGH C.G.  
IXX = .372971E-05  
IYY = .406034E-05  
IZZ = .778220E-05  
IXZ = -.677630E-20  
IYZ = -.338013E-20  
IXY = .952542E-07  
IX/X/ = .293470E-05  
IY/Y/ = .386534E-05  
IX/Y/ = .183227E-06

PRINCIPAL AXES  
OF REFERENCE AXES  
IXX = .130337E-04  
IYY = .406710E-05  
IZZ = .406576E-19  
OF C.G. AXES  
IXX = .400650E-05  
IYY = .371354E-05  
IXY = .169407E-20

OS/

FIGURE 28. NEW DESIGN

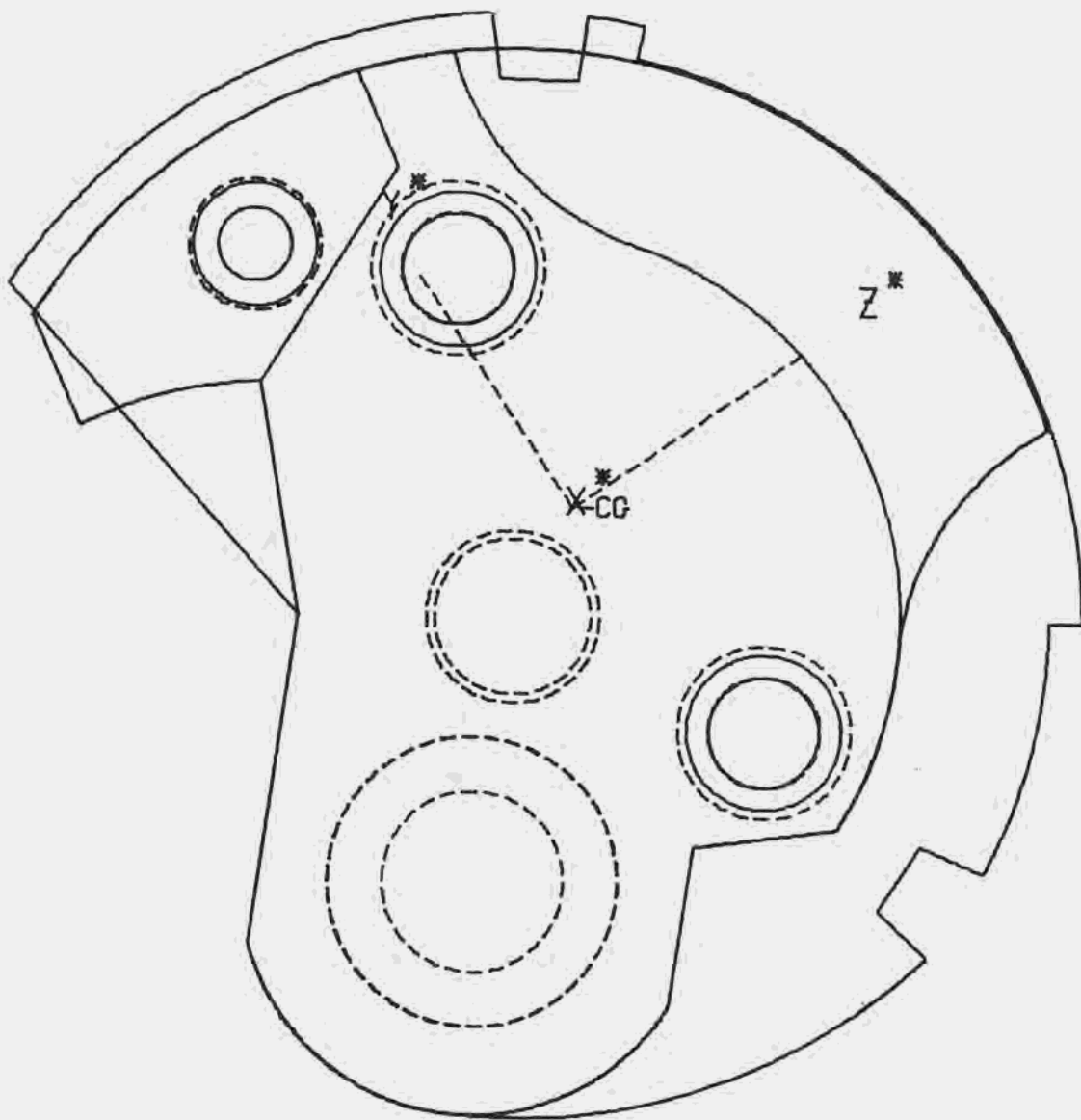
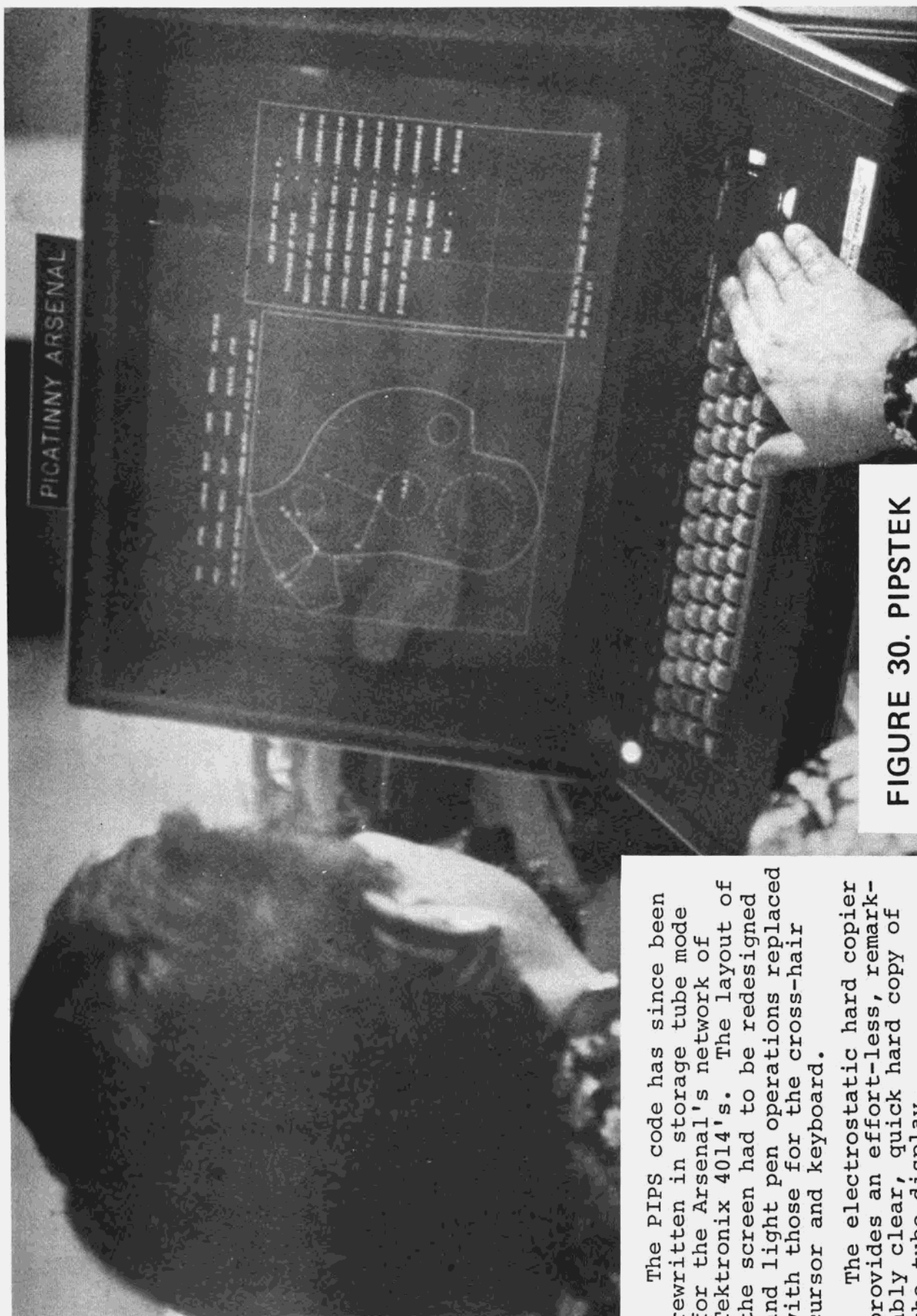


FIGURE 29. HARD COPY FROM PLOTTER



**FIGURE 30. PIPSTEK**

The PIPS code has since been rewritten in storage tube mode for the Arsenal's network of Tektronix 4014's. The layout of the screen had to be redesigned and light pen operations replaced with those for the cross-hair cursor and keyboard.

The electrostatic hard copier provides an effort-less, remarkably clear, quick hard copy of the tube display.

PASS STORE COMBINE PRINT CARDS RETURN NEW PIECE  
 AREA REPLACE RECALL PLOT CHANGE RESULTS STOP

AS NEW REF AXES ORIGIN (----CROSS HAIRS----) AS POINT ON REF X-AXIS

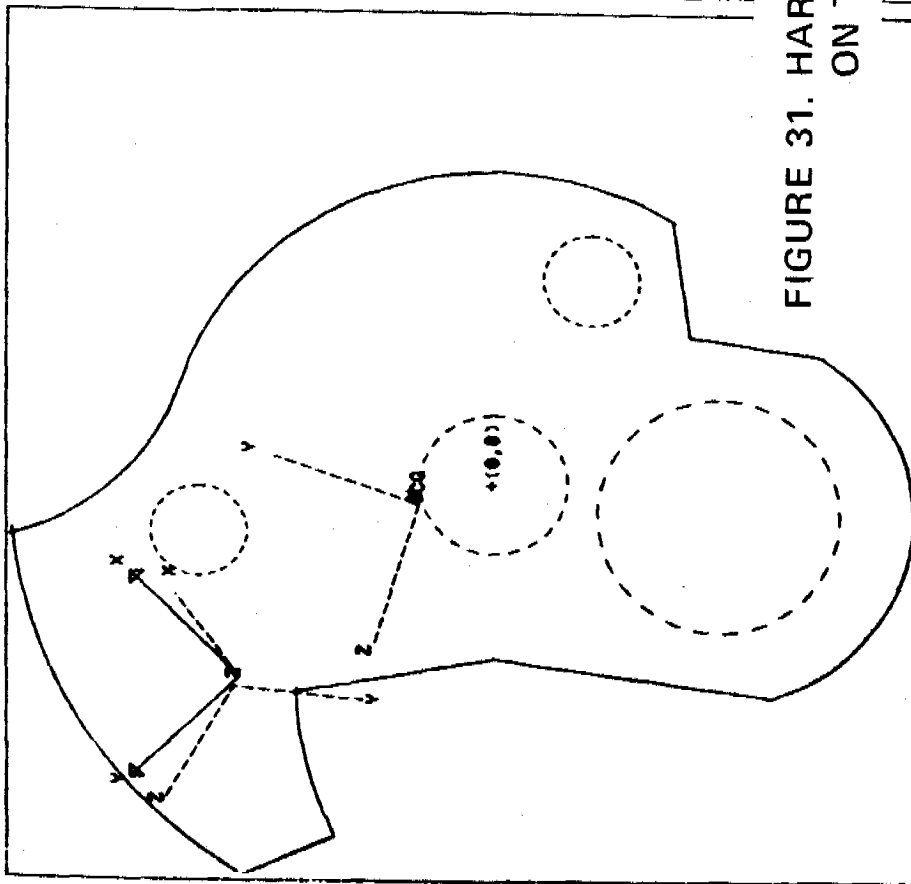


FIGURE 31. HARD COPY OF SINGLE SLICE  
 ON TUBE FACE

GRID SNAP PER INCH = 0.  
 THICKNESS OF PLATE = .100000E+01  
 DENSITY OF PIECE (WEIGHT) = .283000E+00  
 X-COORD USER REFERENCE AXES = -.200000E+00  
 Y-COORD USER REFERENCE AXES = .250000E+00  
 Z-COORD USER REFERENCE AXES = .300000E+00  
 ANGLE(USER REF AXES & HORIZ) = .450000E+00  
 Z-COORD OF MIDDLE OF PIECE = .500000E+00  
 PIECE NUMBER = 1.0000000  
 SCALE = 0.2846100

DO YOU WISH TO CHANGE ANY OF THE ABOVE INPUTS  
 IF SO PICK IT

FIGURE 32.

Hard copy of one page of the library of slices. Selections are made by locating the CRT's cross-hairs over any portion of the item desired.

PICK THE ITEMS YOU WISH, TO HAVE ACCUMULATE FOR MOMENT CALCULATIONS, FROM THE MENU.

LIST

PAGE

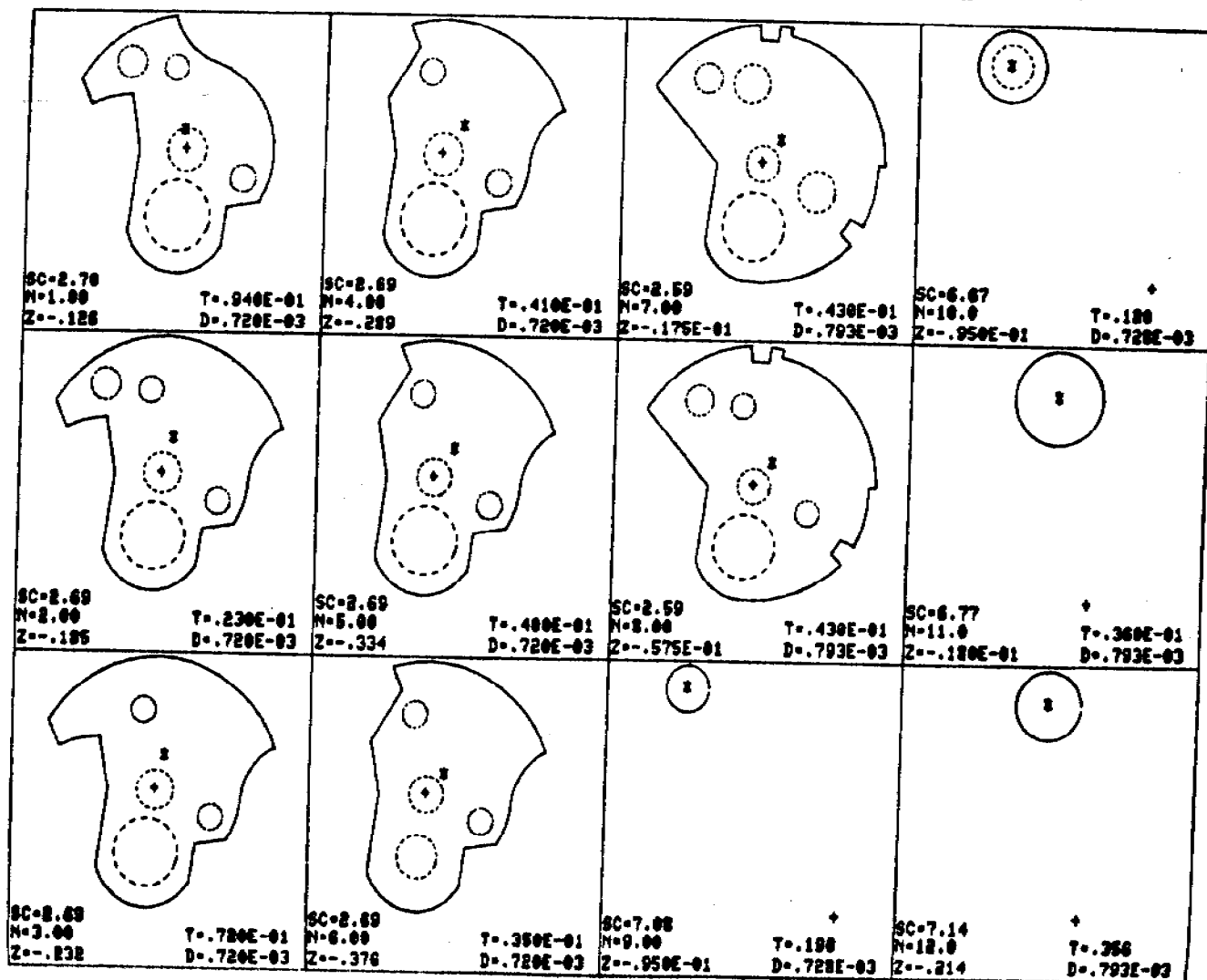
REPAGE

RESTART

DONE

LEAVE

PAGE 1.





#### D. Actual Applications of PIPS Program

The PIPS program has been utilized extensively by the fuze community at Picatinny Arsenal to reduce significantly the man hours and cost incurred in the determination of mass parameters of fuze components. Three typical applications will now be discussed.

##### (1) Sequential Leaf Mechanism of M509 Fuze

###### (a) Description of System

The device depicted in Fig. 33 senses the velocity change from a continued linear acceleration in the direction of the projectile axis. Fig. 34 shows the three leaves and the rotor, which are the principal components of the mechanism. Leaves 1 and 2 (refer to Fig. 33) are held in position by the leaf springs, while leaf 3 is restrained by the anti-reset spring and leaf 2. The spring loaded rotor (spring not shown) which contains a detonator, is maintained in its original safe position by a rotor pin attached to leaf 3. When a sustained acceleration occurs, the first leaf rotates through an angle sufficient to release the second leaf, which after rotating through a given angle, releases the third leaf. When this last leaf completes its rotation, the rotor is disengaged and permitted to rotate to a position where it completes an electric circuit, thus arming the fuze.

###### (b) Problem

Rounds using this fuze have experienced occasional poor performance including premature detonation, which is a critical safety problem. It has been hypothesized that duds have occurred because the leaves have not released the rotor and that premature detonation has been encountered because the rotor has armed too early.

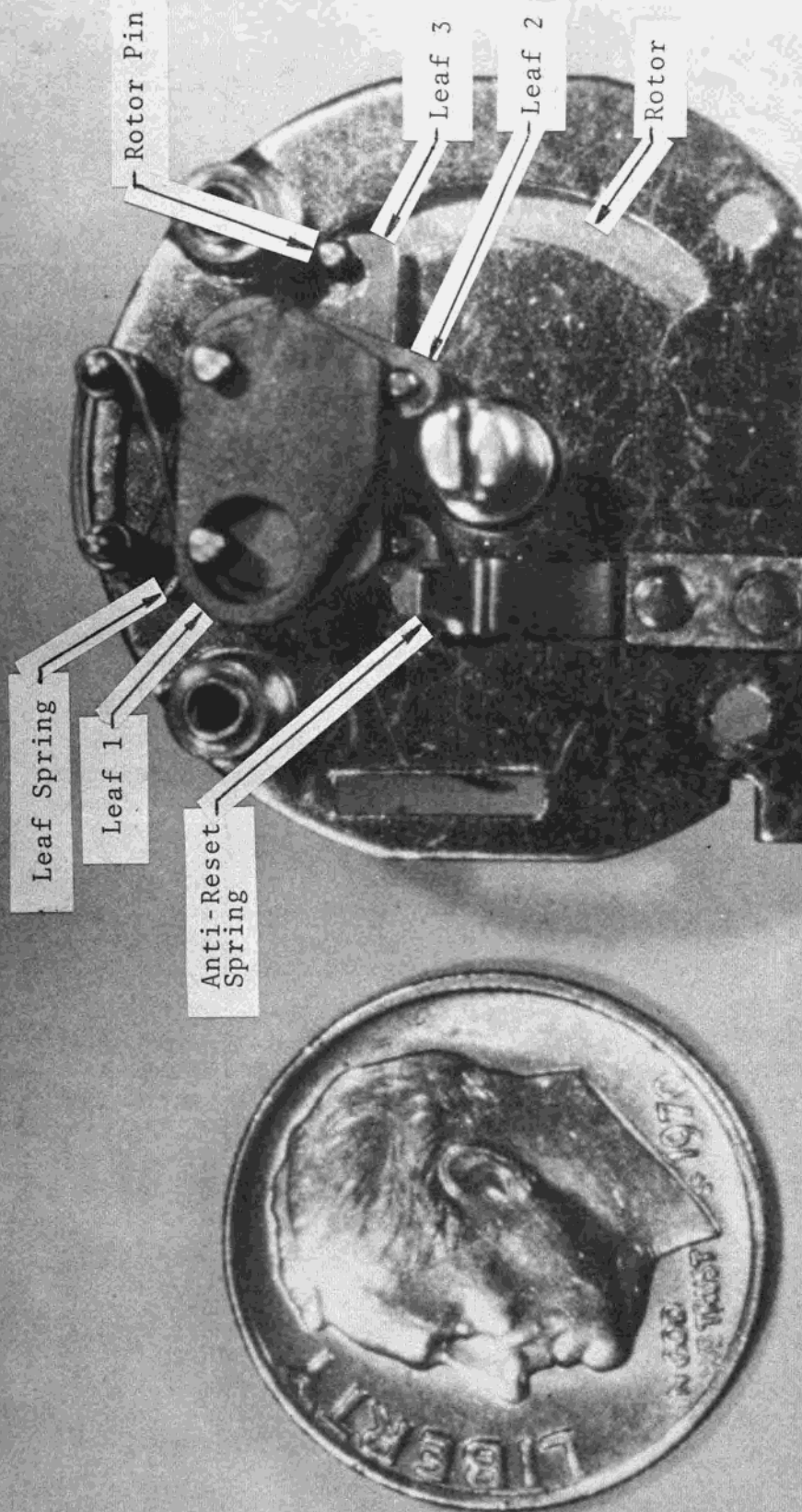


FIGURE 33. M509 FUZE SAFING AND ARMING MECHANISM

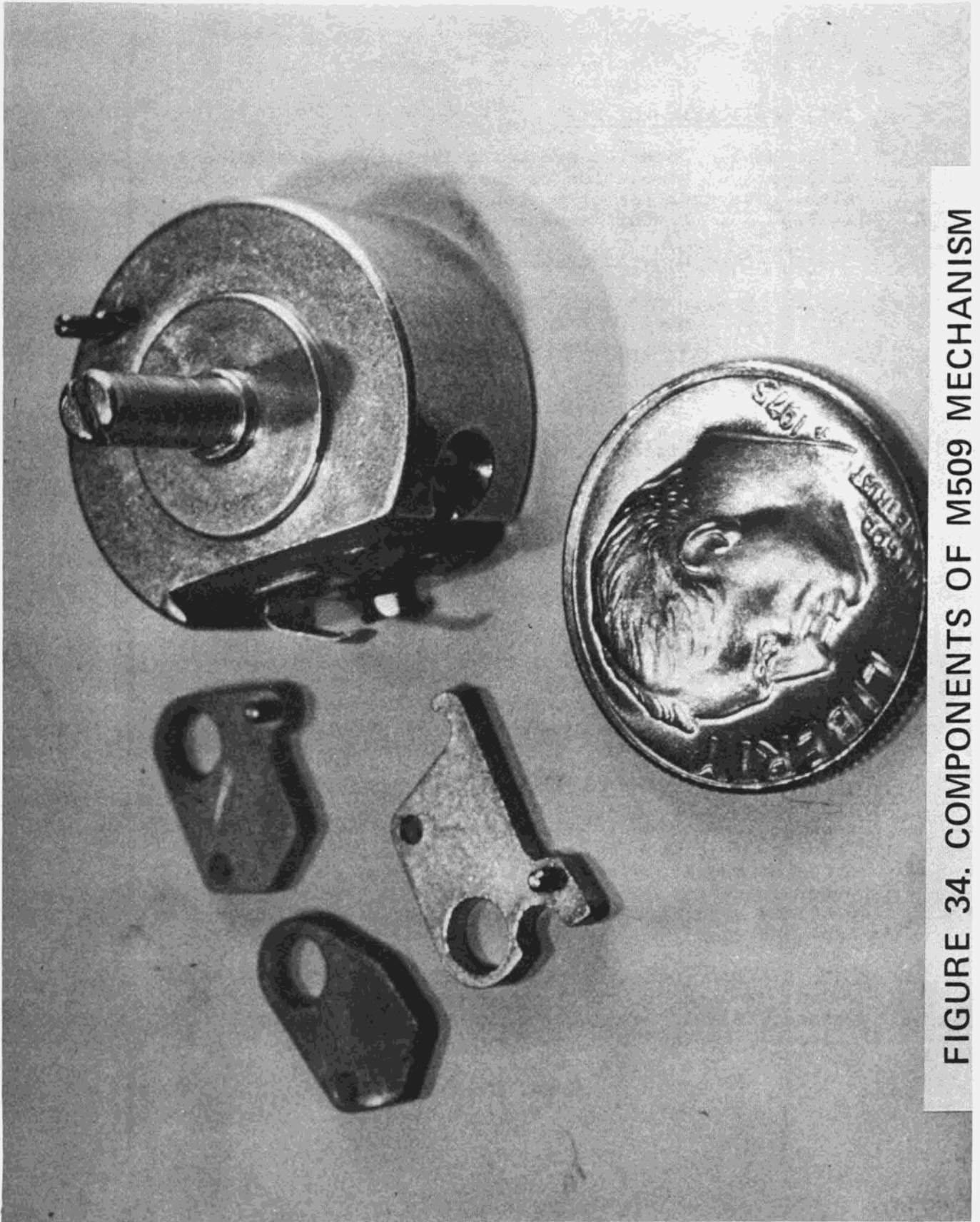


FIGURE 34. COMPONENTS OF M509 MECHANISM

(c) Application of PIPS

These hypotheses are being tested by developing a mathematical simulation of the behavior of the mechanism. The equation of motion for leaf 1 is presented in Fig. 35. In this equation,

$\theta_1, \ddot{\theta}_1$  = angular displacement and acceleration of leaf 1, respectively

$I_1$  = moment of inertia of leaf 1 with respect to its pivot (determined by PIPS)

$m_1$  = mass of leaf 1 (determined by PIPS and verified by weighing)

$r_{c1}$  = distance to c.g. of leaf 1 measured from its pivot (determined by PIPS)

$\alpha$  = initial angular orientation of leaf 1

$\mu$  = coefficient of friction

$\ddot{S}$  = acceleration of projectile

The parameters  $I_1$  and  $r_{c1}$  were determined with the help of the PIPS program. To verify the magnitudes of these parameters, the leaf was weighed and the measured mass compared to that computed by PIPS. The difference was approximately 3%, confirming the validity of the values for  $I_1$  and  $r_{c1}$ . The time involved in obtaining this data for leaf 1 was two hours. Had this been attempted utilizing conventional techniques, it is estimated that a minimum of three days would have been required to derive this information with comparable accuracy. (Of course, additional time of the same order of magnitude would have been required for the other leaves.) Thus, the high level of automation provided by PIPS resulted in a substantial savings in time and cost.

Similar equations of motion were derived for leaves 2 and 3 and the rotor. A computer program has been developed, and is currently being exercised, to discover the cause of the field malfunctions.

## EQUATION OF MOTION OF FIRST LEAF OF SAFING AND ARMING MECHANISM OF M509 FUZE

$$I_{p1} \ddot{\theta}_1 = M_1 r_{c1} \ddot{S} \cos(\alpha + \theta_1) + \text{PIVOT FRICTION TORQUE} \\ + \text{SPRING TORQUE} + \text{FRICTION TORQUE DUE TO SPRING} \\ + \text{TORQUE DUE TO FORCE FROM LEAF 2 EXERTED} \\ \text{ON LEAF 1 (INCLUDING FRICTION)}$$

FIGURE 35.

(2) Proposed Safing and Arming Mechanism of TOW Missile

(a) Description of System

This mechanism is similar to the previous device with the exception that only two leaves are currently used.

(b) Problem

A study was undertaken to evaluate the possibility of replacing the present system with a single leaf. A drawing of one of the three proposed leaves is depicted in Fig. 36.

(c) Application of PIPS

A relationship was derived expressing the "g" level required to initiate motion of the leaf as a function of the leaf geometry and mass properties, leaf and rotor spring torques, forces due to leaf and rotor interaction, and friction. The expression for this g-level is given in Fig. 37. The nomenclature for this equation is as follows:

$F_R$  = force exerted by the rotor on the leaf

$M_R, M_S$  = spring torques on the rotor and leaf, respectively

$W$  = weight of leaf (determined by PIPS)

$X_3$  = location of c.g. (determined by PIPS)

$\mu$  = coefficient of friction

The other parameters are geometric constants which are defined in Fig. 38. (Note:  $x_1 = 0$  for this contour.)

The PIPS program was utilized to locate the position of the c.g. for each of the three candidate designs. It is estimated that approximately one week of engineering time was saved by applying PIPS.

With the help of PIPS, the mathematical simulation was able to demonstrate the feasibility of the single leaf design.

# SINGLE LEAF CONFIGURATION FOR SAFING AND ARMING MECHANISM FOR TOW MISSILE

319

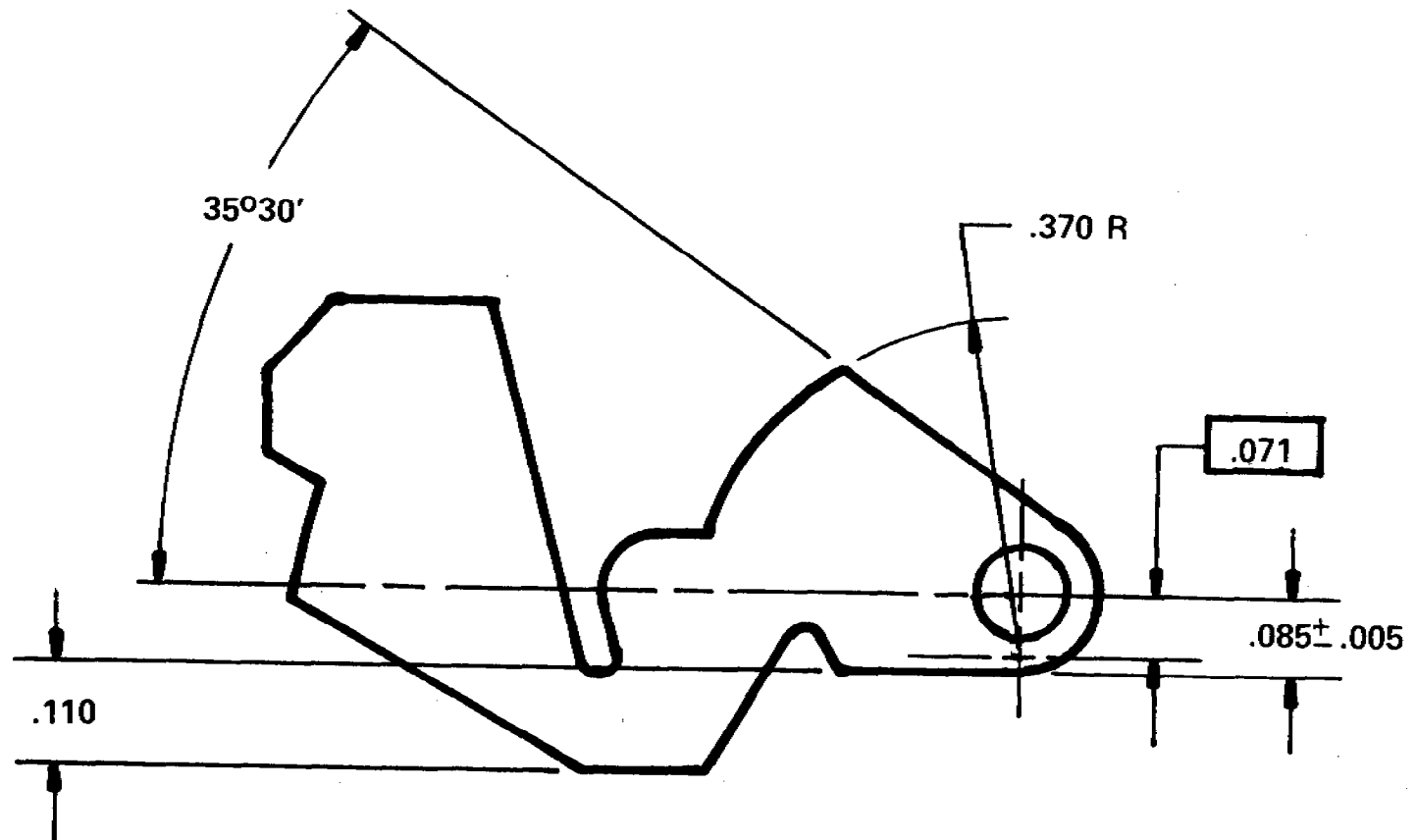


FIGURE 36.

# **G-LEVEL NEEDED TO ACTUATE LEAF OF SAFING AND ARMING MECHANISM OF TOW MISSILE**

$$G = \frac{\left\{ \frac{M_R}{R_2} \left[ \frac{(X_1 - R_1 \cos \beta)(\mu \cos \beta - \sin \beta)}{\cos \beta + \mu \sin \beta} - y_2 \right] - M_S - \mu r_p F_R \right\}}{W(X_3 + \mu r_p)}$$

$$F_R = \frac{(1 + \mu^2)^{1/2} M_R}{R_2 (\cos \beta + \mu \sin \beta)}$$

FIGURE 37.



[illegible]

321

(3) Timing Mechanism of M577 Fuze

(a) Description of System

Two views of the timer of the M577 fuze are shown in Figs. 39a and 39b. Fig. 40 is a schematic diagram of the functioning sequence of this mechanism. In the safe position, the setback pin, which is spring loaded, blocks the path of motion of the spin detent, which is also spring loaded (spring not shown in Fig.40). The spin detent engages the balance wheel of the timer, thus preventing it from operating. Under the influence of the acceleration of the projectile in the gun tube, the setback pin retracts against the setback spring, releasing the spin detent. As the projectile progresses through the gun bore, rotation causes the spin detent to move radially outward against its spring until stopped by the fuze body. This movement frees the balance wheel and sets the timer mechanism into motion, eventually arming the fuze.

(b) Problem

An unacceptable dud rate was experienced during lot acceptance testing of the fuze. It was hypothesized that the setback pin had returned to its original position before the spin detent had released the balance wheel, which, of course, would result in a dud. Such an event is possible since, due to friction torques induced by the projectile acceleration, the spin detent does not move for a considerable portion of the acceleration pulse. It is only when the acceleration level has decayed sufficiently that the spin detent can move and disengage the balance wheel. By that time, the force exerted on the setback pin by the setback spring may be greater than that due to the acceleration. This makes it possible for the setback pin to return and obstruct the path of motion of the spin detent.

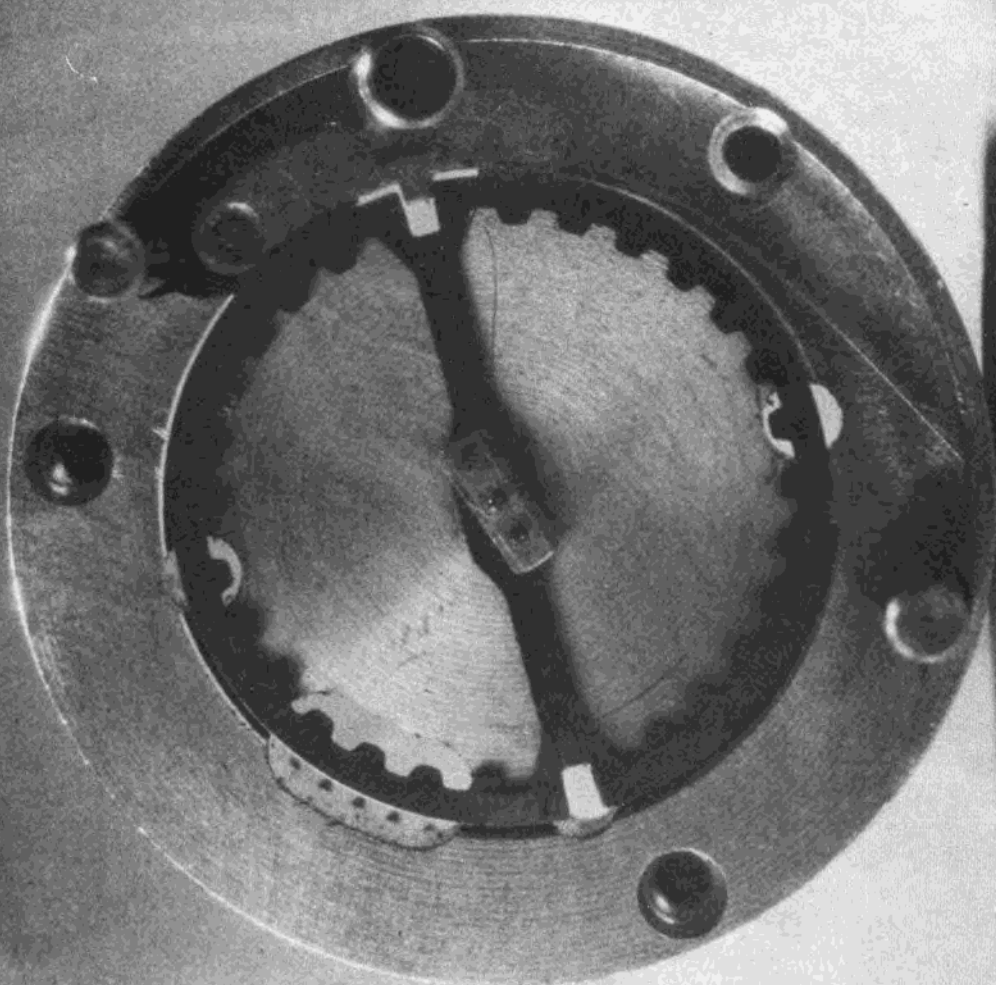
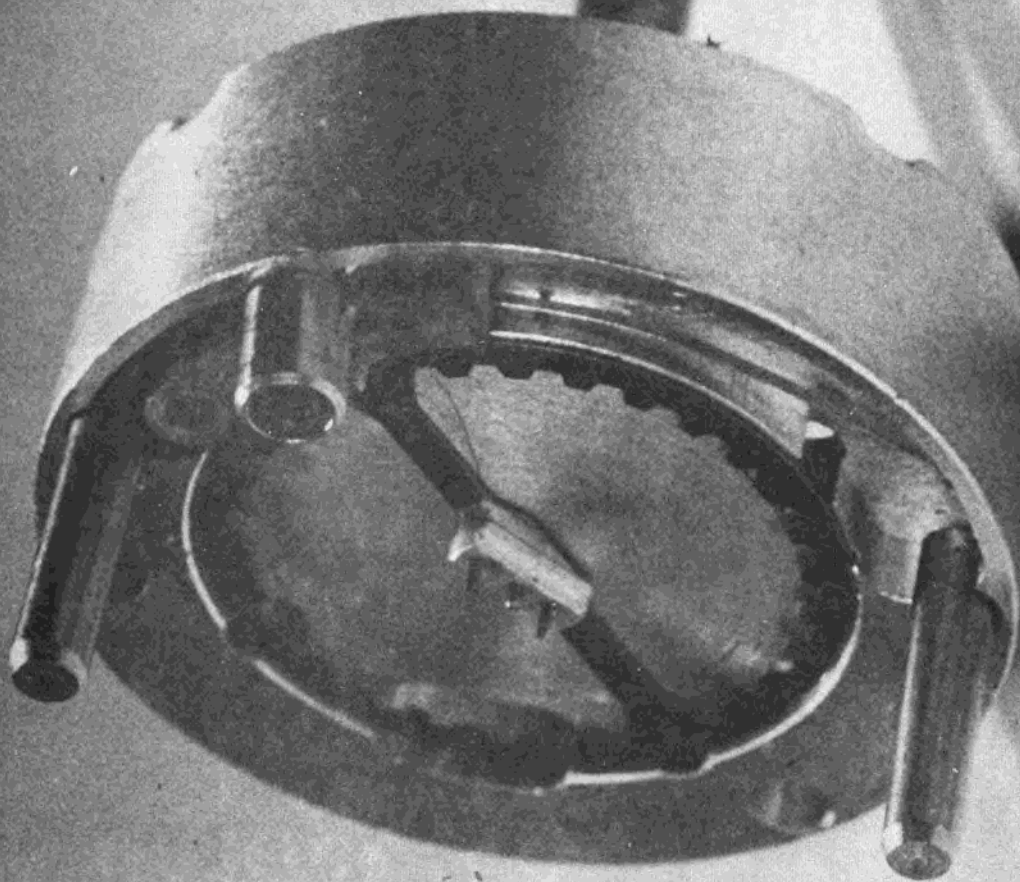
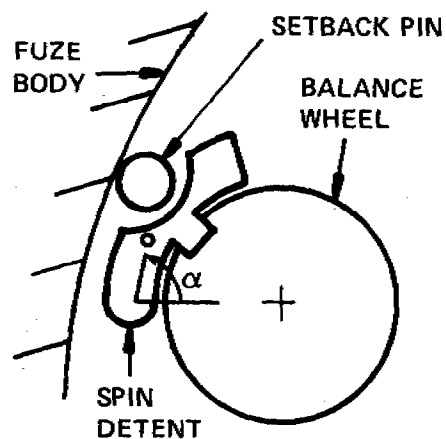


FIGURE 39a. M577 FUZE TIMING MECHANISM

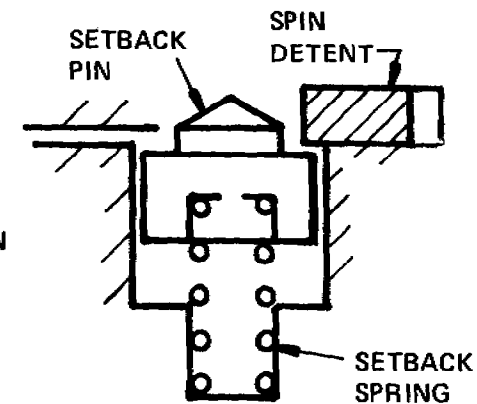
FIGURE 39b. M577 FUZE TIMING MECHANISM



# FUNCTIONING SEQUENCE OF M577 FUZE TIMING MECHANISM

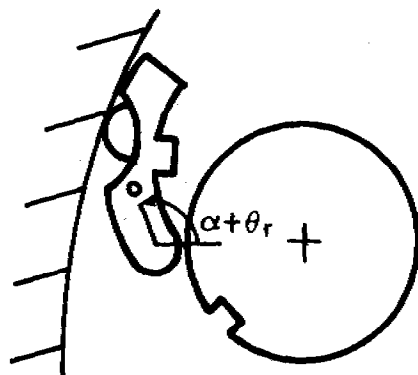


POSITION 1  
BALANCE IS LOCKED  
 $\theta = 0$

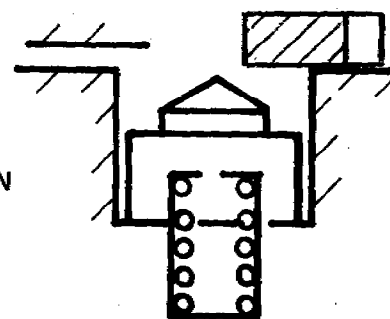


POSITION 1  
REST POSITION AGAINST DETENT

(A)  
SAFE POSITION



POSITION 2  
WALL REACHED  
 $\theta = \theta_r$



POSITION 2  
FULLY RETRACTED

(B)  
ARMED POSITION

FIGURE 40.

(c) PIPS Application

In order to verify or disprove this hypothesis, a mathematical model of the behavior of this system was developed. The differential equation given in Fig. 41 was derived to describe the motion of the spin detent (see Fig. 42). In that equation:

$I$  = moment of inertia of detent (determined by PIPS)

$I_{fr}$  = additional effective moment of inertia due to frictional effects

$\theta, \dot{\theta}, \ddot{\theta}$  = angular displacement, velocity and acceleration of detent, respectively

$M_d$  = mass of detent (determined by PIPS and verified by weighing)

$L$  = distance from projectile spin axis to pivot of spin detent

$r_c$  = location of c.g. of detent with respect to pivot (determined by PIPS)

$\omega, \dot{\omega}$  = angular velocity and acceleration of projectile

$\alpha$  = initial angular orientation of spin detent

The use of the PIPS program to compute the mass, moment of inertia and c.g. location of the detent resulted in a savings of approximately three days of engineering time. (To verify the validity of the calculated values, the spin detent was weighed, and the measured mass differed only negligibly from the calculated mass.)

The analysis showed that the motion of the spin detent is very sensitive to friction; a dud is quite likely to occur whenever the coefficient of friction is of the order of .35. The dud problem was eventually alleviated when the setback pin/setback spring system was modified.

## EQUATION OF MOTION OF SPIN DETENT IN TIMING MECHANISM OF M577 FUZE

$$\begin{aligned}
 (I + I_{fr} \frac{\dot{\theta}}{|\dot{\theta}|}) \ddot{\theta} = & \text{SPRING TORQUE} + \text{PIVOT FRICTION TORQUE} \\
 & + \text{FRICTION TORQUE DUE TO SETBACK ACCEL} \\
 & + \text{TORQUE DUE TO FORCE EXERTED BY} \\
 & \text{BALANCE WHEEL ON SPIN DETENT} \\
 & + M_d L r_c \left[ \omega^2 \sin(\alpha + \theta) + \dot{\omega} \cos(\alpha + \theta) \right] - I \dot{\omega}
 \end{aligned}$$

FIGURE 41.

FIGURE 42. SPIN DETENT





#### E. Development Plans and Conclusions

There are other programs, batch and graphics, that perform similar calculations on different shapes: solids of revolution and asymmetrical systems of bodies (with major axes at skew angles). It is intended to incorporate the special features and strengths of these other programs into the next version of the PIPS code along with the coding to use the tablet as a digitizing device to permit rapid input of contours directly from drawings or lofting templates.

As a final reflection - the development of batch processing computer programs to solve problems of this type has proved inadequate to satisfy the analysis needs of the designers who must use them. The perceptual augmentation that graphics provides has proved to be an essential ingredient in the design - analysis process, and to insure maximum usefulness, a substantial amount of human engineering must go into the development of the graphics procedures.

## DEVELOPMENT PLANS

### COMBINE CAPABILITIES:

- PIPS\* (PLANAR SOLIDS)
- PROMS,\* PHASOR,\* WEIGHT (AXISYMMETRICAL SHAPES)
- SOS, MOMENTS (ASYMMETRICAL SYSTEM OF BODIES)

\* GRAPHICS PROGRAMS

FIGURE 43.



FIGURE 44. DATA TABLET



## TESTING ALGORITHMS FOR A MINI-COMPUTER ON A MAXI

F. D. Crary

The Mathematics Research Center  
The University of Wisconsin-Madison

ABSTRACT. A common problem in the development or testing of algorithms to be run on minicomputers is the lack of access to the eventual target machine. In such cases, development is done on other equipment. Occasionally the success (or failure) of the development effort may directly influence the choice of minicomputer to be used in the application.

An obvious approach to this problem is to simulate the target minicomputer on available large scale equipment. This approach poses certain difficulties. Even if a package of subroutines to simulate the minicomputer arithmetic is accessible from a high-level language, the programming problems are difficult. Since calling upon such subroutines usually amounts to coding in assembly language disguised in the syntax of a high-level language, nearly all of the problems of assembly language coding face the programmer: the programming time is excessive, the programs are difficult to comprehend and correct, etc.

A solution to these difficulties is the use of an appropriate preprocessor. The input to the preprocessor is the desired program expressed in a high-level programming language. The output of the preprocessor is the same program expressed in terms of references to the subroutines simulating the minicomputer. Thus the preprocessor has done the "dirty work" of preparing the disguised assembly language version of the program.

A desirable capability of such a preprocessor is to be able to accept a description of the supporting subroutine package and produce a program according to that description. This can be useful if a number of different simulations must be made--neither the input program nor the preprocessor need be changed, only the description.

Such a preprocessor to extend Fortran (called AUGMENT) is available from the Mathematics Research Center. The input language is Fortran extended by user-defined data types, operators, and functions. AUGMENT's output is an ANSI Standard Fortran program with nonstandard operations replaced by calls to appropriate subroutines. The language extension capabilities of AUGMENT make it suitable for an application needing nonstandard arithmetics. Some other applications are mentioned in the text.

---

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

1. INTRODUCTION. The initial problem that we address is the development of software for minicomputer applications on large scale computers. This development may be required for a number of reasons:

Algorithms may be destined for a number of different computers, more than can conveniently be made available to the developer.

Algorithms may be of varying sophistication for different applications.

The requirements of the algorithm may determine the minicomputer to be used in the application; that is, the algorithm development must precede acquisition of the minicomputer.

The file storage and editing capabilities of the large system may offer an attractive development environment.

Any number of these reasons (and possibly others) may combine to make software development and/or maintenance on large scale equipment desirable or necessary.

An obvious means of performing such development begins with the development or acquisition of a package of subroutines that allows the simulation of the essential aspects of the arithmetic on the minicomputer to be used for the application. The proposed algorithms are then tested in the large machine by coding references to the routines in the simulation package.

This approach to development has several difficulties. First, consider the kind of code that must be written. In Fig. 1(a), we have a well-known algebraic expression. We assume that a straightforward evaluation of this expression is desired (the proper choice of numerical method is a very important subject, but we do not concern ourselves with it in this paper). If we were to evaluate this expression by references to a collection of three argument subroutines, we would obtain code of the form shown in Fig. 1(b). If the simulating routines supported a simulated accumulator, Fig. 1(c) could result. Finally, simulating routines implemented as functions could yield Fig. 1(d).

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

(a) original expression

```
CALL MUL (B, B, TMP(1))
CALL MUL (FOUR, A, TMP(2))
CALL MUL (TMP(2), C, TMP(2))
CALL SUB (TMP(1), TMP(2), TMP(2))
CALL RT2 (TMP(2), TMP(2))
CALL SUB (TMP(2), B, TMP(2))
CALL ADD (A, A, TMP(1))
CALL DIV (TMP(2), TMP(1), X)
```

(b) subroutine package

```
CALL LOAD (FOUR)
CALL MUL (A)
CALL MUL (C)
CALL STORE (TMP(1))
CALL LOAD (B)
CALL MUL (B)
CALL SUB (TMP(1))
CALL RT2
CALL SUB (B)
CALL DIV (TWO)
CALL DIV (A)
CALL STORE (X)
```

(c) simulated accumulator

```
X = QUOT (DIFF (RT2 (DIFF (PROD (B, B), PROD
(FOUR, PROD (A, C)))), B), SUM (A, A))
```

(d) functions

Figure 1.

Code generated by calling on  
simulating packages

The point of Fig. 1 is that the resulting code to be used to test the algorithm is essentially assembly language code. The use of a high-level language and compiler serve only to disguise that fact, and not very well since more code might be written than would be required by a true assembly language program. Programs written in this form suffer from almost all the faults of assembly language programs:

- programming time and effort are excessive
- programs are difficult to comprehend and maintain
- debugging is more difficult
- algorithm testing is more difficult because it is harder to make changes

An additional difficulty can occur if more than one machine is to be simulated in this fashion. If the simulating packages for the various machines are obtained from different sources, they may have different calling conventions. Thus several different versions of the same program may need to be written, debugged, and maintained during the development process.

2. AUGMENT. The Mathematics Research Center has developed the AUGMENT precompiler to solve problems of the sort discussed above. AUGMENT is a flexible preprocessor that extends the Fortran language to include nonstandard data types, operators, and functions. Its operation is summarized in Figure 2. The input to AUGMENT is a source program written in the extension to Fortran, accompanied by a Description Deck.

The Description Deck contains information about the extension to the Fortran language and how the extension is implemented by the simulating package. A portion of a Description Deck with an explanation of its contents is contained in the Appendix.

To test an algorithm for a minicomputer using AUGMENT, one would need a simulating package that supported the minicomputer data types to be used in the algorithm (INTEGER, REAL, and perhaps DOUBLE PRECISION), and a Description Deck for the package. Then the algorithm is written in Fortran with the various quantities declared as nonstandard data types as specified by the Description Deck. A pass through AUGMENT generates a Fortran program which makes calls on the simulating package (see Fig. 3). From that point, compilation, linking/loading, and testing proceed as usual.

AUGMENT eliminates the need to program in disguised assembly language by creating that form automatically. AUGMENT also eliminates the need for several versions of the same algorithm if more than one machine is to be simulated. In this case, only the Description Deck need be changed to change simulated machine and adapt to different calling conventions.



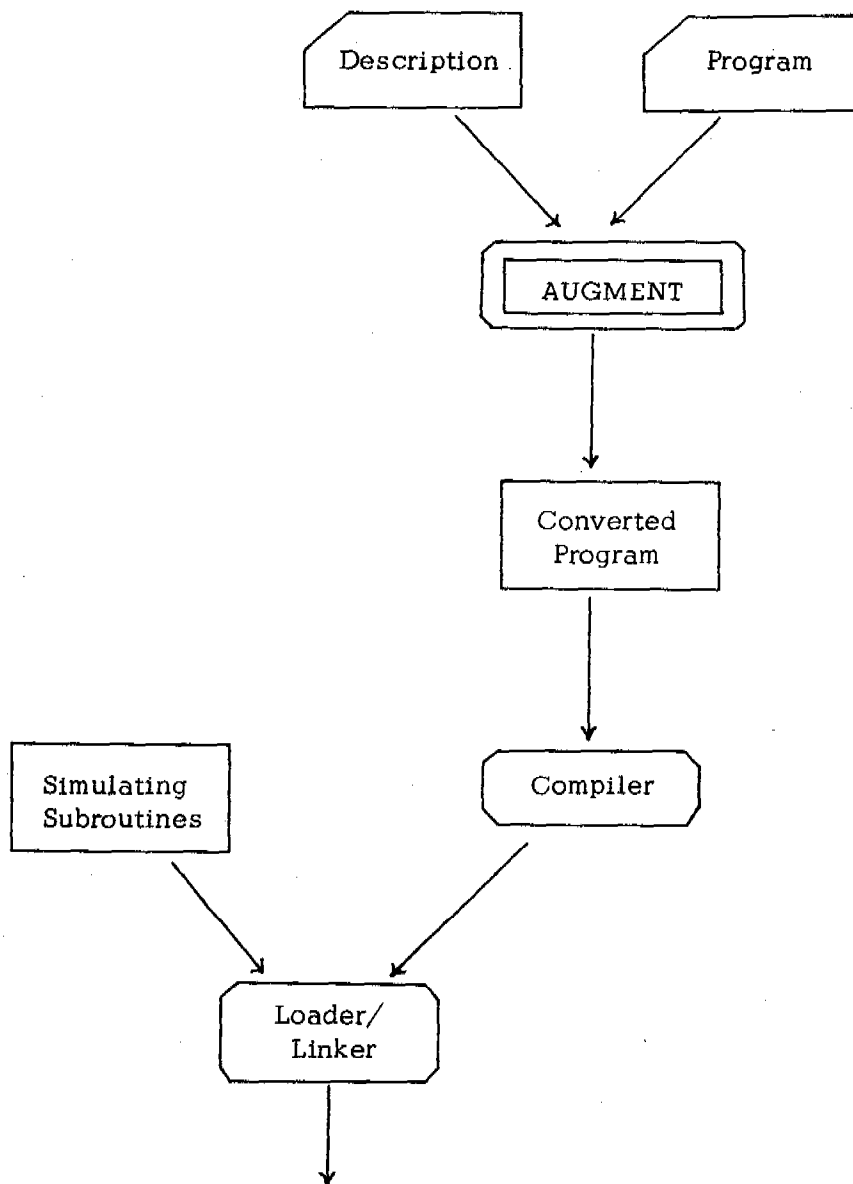


Figure 2.

Summary of AUGMENT use

3. OTHER APPLICATIONS. AUGMENT can be applied in any application where nonstandard data types or operations are needed. Some of these applications are:

- instruction counts. To obtain an estimate of the amount of work performed by a subroutine, one can prepare a simulating package which performs the usual machine arithmetic operations with the side effect of counting the number of operations that are performed. This may, of course, be combined with a minicomputer simulation.
- multiple precision arithmetic. On those occasions when one requires greater accuracy than is provided by the machine, a software multiple precision may be easily employed with the aid of AUGMENT.
- interval arithmetic. Interval arithmetic is a tool for obtaining valid results in spite of the inherent inaccuracies of machine computation. The method computes with closed intervals rather than single numbers. See [ 3 ] for more information.
- Taylor series. In a package under development, a truncated Taylor series is maintained for each variable. When an operation is performed, the Taylor series for the result is computed.

---

AUGMENT input:

$$X = (-B + \text{SQRT} (B*B - 4.*A*C)) / (2.*A)$$

AUGMENT output:

```
CALL NEG (B, TMP(1))
CALL MUL (B, B, TMP(2))
CALL CNV (4., TMP(3))
CALL MUL (TMP(3), A, TMP(3))
CALL MUL (TMP(3), C, TMP(3))
CALL SUB (TMP(2), TMP(3), TMP(3))
CALL RT2 (TMP(3), TMP(3))
CALL ADD (TMP(1), TMP(3), TMP(3))
CALL CNV (2., TMP(1))
CALL MUL (TMP(1), A, TMP(1))
CALL DIV (TMP(3), TMP(1), X)
```

Figure 3.

Sample of AUGMENT translation

4. AVAILABILITY. AUGMENT is currently (March 1976) available for the following systems:

UNIVAC 1100  
IBM 370  
CDC 7600  
Honeywell 635

Copies of AUGMENT have been supplied to persons intending to implement it on the following systems:

MULTICS  
DEC PDP-10

Portability of AUGMENT has been found to be very good (in one week, both the IBM and CDC versions were brought up). The precompiler is written in ANSI Standard Fortran except for

- (1) eight machine dependent primitive routines which require about 150 lines (total) to implement, and
- (2) some minor oversights which are documented in [ 2 ].

5. CONCLUSION. The AUGMENT precompiler is a tool which simplifies the programming process when nonstandard arithmetics and data types are required. The applications for AUGMENT include simulation of minicomputers, word length sensitivity analysis, multiple precision arithmetic, operation counting, and interval arithmetic.

AUGMENT will not cure all world's programming problems, but can be of great assistance in dealing with some problems. One user of AUGMENT put it this way: "AUGMENT isn't a program that you need all the time--but when you need it, you really need it."

6. REFERENCES.

1. Crary, F. D., The Augment Precompiler, I. User Information, The University of Wisconsin-Madison, Mathematics Research Center, Technical Summary Report #1469, December 1974.
2. Crary, F. D., The Augment Precompiler, II. Technical Documentation, The University of Wisconsin-Madison, Mathematics Research Center, Technical Summary Report #1470, October 1975.
3. Moore, Ramon E., Interval Analysis, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1966.

References [ 1 ] and [ 2 ] are updated as changes are made to AUGMENT. Current versions are available on COM microfiche from the author.

## APPENDIX

This Appendix presents a portion of a Description Deck for AUGMENT with a brief explanation of its contents. A portion of a Description Deck for a double precision complex data type is presented in Figure 4. The line numbers in the Figure are added for easy reference from the text. The explanation below is by line number.

Line 1. The first line of the description of a data type specifies the name of the type. A program translated under this description may use the data type name DBLCOMPLEX in the same way that INTEGER, REAL, etc. are used in a standard Fortran program. In the lines following, the character "\$" is an abbreviation for "DBLCOMPLEX". This abbreviation is built into AUGMENT to reduce the size of the Description Deck.

```
1. *DESCRIBE DBLCOMPLEX
2. DECLARE DOUBLE PRECISION (2), KIND SAFE SUBROUTINE,
   PREFIX DPC
3. OPERATOR - (NEG, UNARY, PREVIOUS, $)
4. OPERATOR + (, NULL UNARY, PREVIOUS, $)
5. OPERATOR + (ADD, BINARY 1, PREVIOUS, $), - (SUB),
   * (MUL(SUBROUTINE)), / (DIV(SUB))
6. OPERATOR .EQ. (EQ, BINARY 2, PREVIOUS, $, LOGICAL),
   .NE. (NE)
7. FUNCTION CLOG (LN, ($), $), LN ( )
8. CONVERSION CTDC (CFI, INTEGER, $, UPWARD), CTDC (CFR, REAL)
```

Figure 4.

Portion of Description Deck for  
Double Precision Complex Arithmetic  
(Line numbers are not part of the  
Description Deck.)

Line 2. The second line gives the general information about the data type and its supporting package of subprograms. The first clause (DECLARE DOUBLE PRECISION (2)) specifies the way in variables of type DBLCOMPLEX are to be represented in the translated program. A declaration of the form

DBLCOMPLEX A, B(10)

would be translated to the form

DOUBLE PRECISION A(2), B(2,10)

The second clause indicates the nature of the supporting routines. Three choices are available: function, subroutine (three argument), and simulated accumulator. The clause SAFE SUBROUTINE specifies that the routines are three argument subroutines and that it is safe to allow the output argument to be one of the operand arguments. This specification may be overridden for specific routines (see Line 5). The last clause (PREFIX DPC) specifies the manner in which the routines are named. Unless overridden, all routine names will begin with the letters "DPC" as will the names of all temporary locations generated by AUGMENT.

Line 3. Beginning with Line 3, we describe the operators that may have DBLCOMPLEX operands. Line 3 describes the unary minus (negation) operator. It is implemented by the subroutine DPCNEG (the "DPC" is not written since it is the prefix--AUGMENT adds it). The remaining information in the line specifies that the operator is unary, that its position in the operator hierarchy is previously defined (the hierarchy position is attached to the symbol used in the source program), and that this description of the operator is for operands of type DBLCOMPLEX (the result is also of type DBLCOMPLEX since unary operators do not change type).

Line 4 describes the unary + operator. Since the description contains the phrase "NULL UNARY", AUGMENT will discard all unary + operators with an operand of type DBLCOMPLEX. Since it is to be discarded, no routine name need be specified.

Line 5. Next come the arithmetic operators. The notation "BINARY 1" specifies that the operator being described is binary (has two operands) and that both operands and the result for this description have the same type (DBLCOMPLEX). This line also indicates the "drop out" rules allowed by AUGMENT. The subtraction operator has exactly the same description as the addition operator except for subroutine name, hence only the subroutine name need be specified. For the multiplication and division operators, the routines are such that it is not safe for the result argument to be one of the operand arguments. Hence the "SAFE SUBROUTINE" specification of Line 2 is overridden by the specification "SUBROUTINE" which may be abbreviated "SUB".

Line 6. The relational operators .EQ. and .NE. are also meaningful for complex types. The "BINARY 2" specification means that the description will contain two data type names: the first is the type of the operands; and the second, the type of the result. Since none of the other relational operators are defined for DBLCOMPLEX operands, their use will cause AUGMENT to diagnose an error.

Line 7. Next we describe two of the functions which may operate on DBLCOMPLEX operands. Functions for AUGMENT are generic. The logarithm function is described as being implemented by the routine DPCLN when it appears with the argument list "(\$)", (that is, when it appears with a single argument of type DBLCOMPLEX), and yields a result of type DBLCOMPLEX. This function may also be called by the name LN with the same description.

Line 8 describes two conversion functions. Conversion functions are applied automatically in certain cases of mixed mode operands or may be coded in the program. The function CTDC (Convert To DbComplex) is defined for INTEGER and REAL arguments with DBLCOMPLEX results. In each case, the conversion is "upward" meaning that DBLCOMPLEX is the preferred type if automatic conversion is applied. That is, the expression "I+D" is equivalent to "CTDC(I)+D" if I is INTEGER and D is DBLCOMPLEX. "Downward" conversion functions are applied automatically only across the replacement operator.

Other features and options are available which are not illustrated by this example. The user documentation [1] describes these cases.

USE OF A MINI-COMPUTER FOR ON-LINE REAL-TIME  
PROCESSING OF MASS SPECTRAL DATA FROM MULTIPLE MASS SPECTROMETERS

D.H. Robertson and C. Merritt, Jr.

U.S. Army Natick Research and Development Command

Natick, Massachusetts 01760

ABSTRACT

One of the greatest challenges in the area of automated data processing is the output from a mass spectrometer; combined gas chromatography/mass spectrometry provides still more stringent demands on the data processing system. In order to realize the goal of on-line real-time acquisition and reduction of mass spectrometry data from multiple instruments, special means were developed for this purpose. As an adjunct to acquisition and reduction processes, there exists the problem of identification of the sample by means of library search routines which must be rapid to be effective.

Several mathematical manipulations were conceived and evaluated on the raw data to facilitate identification; library file structure was investigated as a primary means effecting this. These approaches to identification and the library file structure are discussed. Typical of the questions which must be answered in the latter case are: What are the features of an optimum library search algorithm and the optimum library size; and is it reasonable to combine pattern classification techniques with a library search? Also the computer system, a DEC 15/76, legitimately in the mini-computer class, is described; especially as it relates to the automation of an analytical chemistry laboratory.

The primary function of the computer system in the Analytical Chemistry Laboratories at the Natick R&D Command, has been to automate the instrumentation used to perform chemical analyses. The most challenging aspects of automation are presented by on-line real-time acquisition, reduction and interpretation of mass spectrometer output. In the area of interpretation, several mathematical data treatments, in succession, have been developed which have been used in sequence as the size of the data base has increased. The hardware and software specifications of our system are described herein followed by a review of the mathematical treatments and presentation of some applications of the overall system.

The basic system is a PDP15/76, manufactured by Digital Equipment Corporation. It is equipped with 48K of core memory in the central processor configuration and 2 PDP11 memories respectively, controlling the UNIBUS access to the 15 memory and the I/O operations for the GT40, a graphic display system. The 15 memory directly controls a fixed head disc with a 250,000 word capacity i.e.,  $2^{18}$  (262,144); a cartridge disc system operates through the peripheral processor, which allows the 15 memory to access the 11 memory associated with the unibus.

Figure 1 depicts the basic units of our system as well as some of the experiments which are interfaced with the system. The slave computer is a HP 2116B which was the original venture into computer technology in the Analytical Chemistry Laboratories. Simply because the acquisition and reduction routines were fully developed for this disc-oriented system, it has been kept in the link, handling output from a medium resolution mass spectrometer, the output of which is digitized directly with a PAD (pulse amplifier discriminator).



The TOF Mass Spectrometer is equipped with a CSI 260 high speed digitizer as a front end prior to direct data transmission to the 15/76 system. The high resolution mass spectrometer is similarly front-ended with a high speed digitizer unit custom built for the Laboratories by Adage.

It is possible to effect compound identification from first principles, an approach whereby the structure of each unknown is assigned as a function of characteristic mass peaks in a spectrum which refer directly to the presence of specific bonding and/or chemical functionality. As such, a library is not necessary.

In most cases, however, the identification of an unknown compound from its mass spectrum by automatic data processing assumes the existence of a file of data for a large number of known compounds and the capability for searching that file in a manner which will provide component identification. The conventional approach uses tables of mass vs. intensity values that constitute digitized mass spectra. By comparing the unknown to each known spectrum in the library, it is possible to achieve identification from the best match of the unknown to a known spectrum in the library file.

Because of the large number of spectra to be searched in a typical application, that of combined gc/ms operation for instance, the computer configuration required for identification of unknown based on use of all their mass and intensity data i.e., the complete spectrum, is normally

quite large and counter to the best interests of efficient on-line real-time handling of mass spectral data. In particular, there is a need in the average analytical laboratory for searches on small, relatively inexpensive computers which remain inexpensive only if their peripheral units are confined to modest size and capability.

Historically, three methods were developed for encoding spectra as compressed data:

1. Calculation of entropy function
2. Calculation of divergence
3. Selected binary encoding

The first two approaches to the classification of mass spectral data, namely the calculation of the Khinchine entropy function and of the divergence function, are derived from set theory and are based on the statistical distribution of peaks in a mass spectrum. These as well as the selected binary encoding which is described below, have in common the reduction of the mass spectrum to a single-valued number which is diagnostic within some range for the compound with which it is associated. (Fig. 2.)

The entropy function is calculated by summing the products of individual ion abundances  $p$  and their respective logarithms. " $p$ " thus represents the ion abundance in terms of the percent of total ionization of the molecule in question or in another sense, more germane here, the probability of occurrence of that ion fragment in the spectrum.

Mass spectra are thus converted to a single valued number and in this way a data file can be constructed consisting of these numbers. An example is shown in Table 1. These compounds have been selected to show the typical variation in the "entropy" value which is expected for the

variation in the degree of unsaturation in the molecule. In searching a file of precalculated Khinchine values, a matching index is used to establish correspondence of the value for an unknown with the library value. In the early work with Khinchine values, the construction of the reference file was limited to a few hundred compounds which are normally encountered in the analysis of the volatile components of natural products, most especially of foodstuffs.

When applied to large libraries, in the order of 7000 compounds, the range of values of the Khinchine function was not sufficiently unique. An example of this situation is shown in Table 2.

To provide differentiation between two compounds, the Khinchine functions of which are too nearly alike to be diagnostic, the second function on our list; namely, the divergence calculation was invoked.

The formula for this calculation appears in Figure 3. Here  $N_1$  and  $N_2$  represent, respectively the total number of ions in each of the 2 compounds being compared and "p" has the same meaning as in the Khinchine function calculation. In practice it has been found convenient to refer the calculation of divergence of a given compound in the aliphatic hydrocarbon series for instance, to the normal straight- chained alkane of the same carbon number. In Table 3 are listed the divergence values for several  $C_6$  hydrocarbons referred to N-hexane. For example, in the case of two compounds such as hexene and methylpentene, the divergence values are markedly different.

Considering the fact that the final format in which data are handled in a digital computer is in the binary world we were led to develop an

approach which we call selected binary coding. This code allows compression of the library file through selective binary coding of characteristic peaks and use of variable length logical records.

The coding procedure is illustrated in Figure 4. A hypothetical mass spectrum is shown with a representation of a 16 bit computer word at the top. Selective binary coding is accomplished by dividing the mass range of interest into multiple groups of seven. The number corresponding to the peak in the spectrum in each group which has the highest intensity is then coded as a three bit binary number. In this example the fourth position is encoded in the first grouping, the seventh in the second and so on; zero is used to denote the absence of a peak within the grouping, thereby giving a total of eight possible values, hence the term octal coding by which we have designated this scheme.

Representation of an octal number within the computer requires three bits; thus, in a 16-bit machine such as the HP 2116B used in setting up this system, five octal characters can be stored in each computer word with one bit left over. A single computer word is capable of storing information which spans a range of 35 amu. Compounds with a greater mass range require additional computer words. As many as needed are used; the last word is designated by setting a flag in the 16th bit. A further illustration of this system of encoding is shown in Table 4.

Since 15 bit positions allow  $2^{15}$  or 32768 unique representations, it is possible to encode that many compounds in a unique manner.

The octal code for a mass spectrum may be easily obtained from digitized mass and intensity data acquired on-line and stored for subsequent

processing. A flow chart of the procedure is shown in Figure 4. A normal sequence of data processing involves the following on-line operations.

1. Conversion of analog output to digital i.e., mass and intensity data.

2. Condensation of the data to octal format utilizing the routine in this slide, thus providing the binary equivalent of the "spectrum" for which the library is being searched.

3. When binary "1" is sensed in the 16th bit, the number of words to encode the "spectrum" is known, then it is not necessary to search the entire library but only the subset or sub-library collection which requires that number of words for coding.

The subdivision into subfiles of variable record length and creation of a name file were designed to make maximum use of random access mass storage devices.

For each unknown compound being searched, a matching index is calculated; the five best matches are printed out, thereby anticipating the possibility for identical or nearly identical matching indices. The usefulness of selective binary coding for search and retrieval from large data files has been well illustrated in the literature and for this purpose it actually matters little what size window one uses. If, however, one is to make use of the diagnostic information which is contained in a mass spectrum, the size of the window and the starting masses for each window become vitally important. Consider Table 5. The vertical column at the left refers to the number of the window under

consideration and the horizontal sequence across the top of the table consists of the octal digits below each of which appear the masses corresponding to the octal code. The first window, for example, will contain the masses coded octally from 23 to 29. A series of masses corresponding to a series of ions such as 29, 43, 57 and 71 etc., represents alkyl fragments and will each be encoded as the 7th digit in an odd-numbered window.

A pattern of codes is seen in Table 6 which demonstrates the recurring octal digit pattern for various series of ions which correspond to several common functional groups. The hydrocarbon fragments will always be coded in odd-numbered windows, and when the skeletal structure is a saturated aliphatic hydrocarbon, one would expect the most abundant peaks to be a series of alkyl fragments which would be coded as octal 7. If the compound is unsaturated, the octal code would be a 5. Moreover, ions which are still more unsaturated, such as the alkynyl series having the formula  $C_nH_{2n-3}$  and consisting of the series 39, 53, 67 etc., will be represented by octal digit 3 in odd-numbered windows. It should be noted that oxygen containing peaks such as those arising from ketones and aldehydes are isometric with the alkyl ion fragments and will be coded in octal 7 in odd-numbered windows. The code pattern, which occurs in the even-numbered windows and produces the series of ions which is characteristic for alcohols etc., is seen in the bottom left row of the table.

Additional octal code patterns which are characteristic of functional group type compounds are seen in the right hand column. Thus, the

appearance of a particular digit as a series in an even-numbered window may be correlated with a particular structural feature. Generally speaking, octal patterns from the odd-numbered windows are less specific, whereas the patterns for ions encoded in even-numbered windows are more closely related to functionality. In order to achieve the accurate identification for an unknown spectrum as created on-line from a GS/MS run, it is necessary to use a combination of odd and even numbered windows.

Some examples taken from real data coded from compounds selected at random from a file of mass spectral data are seen in Table 7. The top left shows some patterns for alcohols. Compounds A-D correspond to ethanol, isopropanol, n-propanol and 2-ethyl-1-butanol. The first column of digits corresponds to the first odd-numbered window containing masses 23-29. The successive odd-numbered windows contain, for all the compounds, digits representing hydrocarbon series of fragment ions. The corresponding even-numbered windows all contain the octal digit 2 which is characteristic of rearrangement ions for alcohols and other oxygenated species. In the case of the isopropanol and ethylbutanol the octal digit 5, rather than 7, appears, suggesting the possibility of branching. The 6 which appears in the 3rd window for compound C is an error in coding; normally a 7 would be expected to appear.

At the top right patterns for various esters are shown. As with alcohols the octally encoded digit in the odd-numbered windows is a 7 which corresponds to the series of alkyl fragment ions. In the even-numbered windows the 2s appear, representing oxygenated species. In this case the pattern at the end of the code is found to be diagnostic for esters.

The pattern for aldehydes is similar. Again, sevens occur in odd-numbered windows representing alkyl fragments. In this case there is a contribution to these ions from the isomeric ion containing CO. Compounds C and D are actually 2 different spectra for 3-methylbutanal. These again suggest the possibility for detecting branching by the appearance of 5 in one of the odd-numbered windows. The succession of 1s is found to be characteristic for both aldehydes and ketones.

Most important in the overall is the success of this technique of coding when used with real data as obtained on-line and in real-time from a standard GC/MS system. Moreover, experience with the technique adds further support to the choice of coding one mass in 7 as opposed to one mass in every 14 amu. This choice provides greater success in identification of functional group character when the compound for which one is searching is not in the library.

The most demanding aspect of the operations within the Analytical Chemistry Group is associated with monitoring the volatile components in irradiated-preserved foods. A study has already been completed to indicate that there are no toxic substances produced during this preservation process; the current study of volatile components is being conducted in conjunction with an animal feeding study to determine the alteration in nature, if any, of the nutritional quality of the food stuff. As such, large numbers of samples are produced from the rapid scanning mass spectrometric monitoring of the eluent from a gas chromatograph, by means of which the volatile components are separated subsequent to vacuum distillation into rough fractions based on b.p. of component.



There are two aspects of computer applications to this work. The first is the automatic acquisition, reduction and interpretation of mass spectrometric data based on the principles of octal codification which have been presented earlier. The repetitive nature of this application allows high accuracy in identification inasmuch as one is looking for differences among samples which relate to nutritional value.

The second aspect of this deals with statistical analysis of the quantitative data which are produced from the studies, i.e., a study of the variation in quantitative values for the various components determined as a function of sample treatment, method used for irradiation and storage time. It is customary to plot time of storage after irradiation vs. concentration of component(s) for the various procurements or lots of meat, thereby indicating scatter of the data. Included in this plot are usually the  $\bar{x}$  and 2 sigma lines for each collection of data. The previously described investigations have been conducted on output from low resolution mass spectrometers. At the time of this writing the addition of high resolution capability is nearing completion; it is based on electrical processing of the signal from a CEC 21-110 high resolution mass spectrometer operating via a high speed digitizing unit which was custom built for these laboratories by Adage. The Adage device interfaces through traditional links, directly with the computer where the data are processed with standard routines to provide exact mass values.

Although in-house requirements have not yet dictated the interface of our standalone gas chromatographs to the system, the multi-programming aspects of the PDP 15/76 system will readily allow the addition of many slow devices such as these, liquid-liquid chromatographs and standard spectroscopic devices, viz I.R., U.V. and visible spectrophotometry.

TABLE 1

<u>Compound</u>	<u>Khinchine Function</u>
n-butane	0,926614
2-methylpropene	0.812125
t-2-butene	1,041750
3-methyl-1,2-butadiene	1,215960
1,3,5-hexatriene	1.339620
1,5-hexadiyne	1,605300
3-heptyne	1,379110

TABLE 2

Use of Divergence to Resolve Non-unique Khinchine Functions

<u>Compound</u>	<u>Entropy</u>	<u>Divergence</u>
2,4-hexadiene	0,473	10,97
3-methyl-1,3-pentadiene	0.474	26,19
2-ethyl-1,3-butadiene	0,497	7,79
1,3-hexadiene	0,498	9.17

TABLE 3

Reference Compound for the Series is Hexane

<u>Comparison Compound</u>	<u>Divergence(J)</u>
n-hex-1-ene	4.2269
2-methylpent-2-ene	7.7805
cyclohexane	10.0238
3-hexyne	11.8302
2-methylpentene	12.5054
1-hexyne	18.7163
2-hexyne	25.8092

TABLE 4

Example of Octal Coding

<u>Mass ranges</u>	<u>m/e to be encoded</u>	<u>Position of m/e in octet</u>	<u>Binary Code</u>	<u>Octal Code</u>
23-29	24	2	010	2
30-36	32	3	011	3
37-43	43	7	111	7
44-50	0	0	000	0

TABLE 5

Array of Masses According to Position in Spectrum Grouping\*

Group Number	Position in Group**						
	1	2	3	4	5	6	7
1	23	24	25	26	27	28	29
2	30	31	32	33	34	35	36
3	37	38	39	40	41	42	43
4	44	45	46	47	48	49	50
5	51	52	53	54	55	56	57
6	58	59	60	61	62	63	64
7	65	66	67	68	69	70	71
8	72	73	74	75	76	77	78
9	79	80	81	82	83	84	85
10	86	87	88	89	90	91	92
11	93	94	95	96	97	98	99

\*Circled masses indicate characteristic ion series.

\*\*See Figure 6.

Table 6 Octal Code Patterns Generated by Characteristic Ion Series

ALKYL IONS								ARYL IONS						
o d d	m/e	15	29	43	57	71	etc.	e* v c n	m/e	30	44	58	etc.	
	code	7	7	7	7	7		n	code	1	1	1		
ALKENE & CYCLOALKYL IONS								SULFUR IONS						
o d d	m/e	27	41	55	69	83	etc.	e v c n	m/e	33	47	61	75	etc.
	code	5	5	5	5	5		n	code	4	4	4	4	
ALCOHOL & ETHER IONS														
e v c n	m/e	31	45	59	73	etc.								
	code	2	2	2	2									

\*Indicates odd or even numbered groups (see Table 2).

TABLE 7

PATTERN FOR ALCOHOLS				PATTERN FOR ESTERS			
Compd,				Compd,			
A	7272			A	7202	03	
B	5272	02		B	7271	0203	
C	7262	72		C	7272	04620	3
D	7272	52		D	7272	04620	3
PATTERN FOR ALDEHYDES							
Compd,							
A	7131	7					
B	7271	7071					
C	7251	71717	1				
D	7051	71707	1				

FIG. 1

# FOOD SCIENCES LABORATORY

## Mass Spectral Data Acquisition System

359

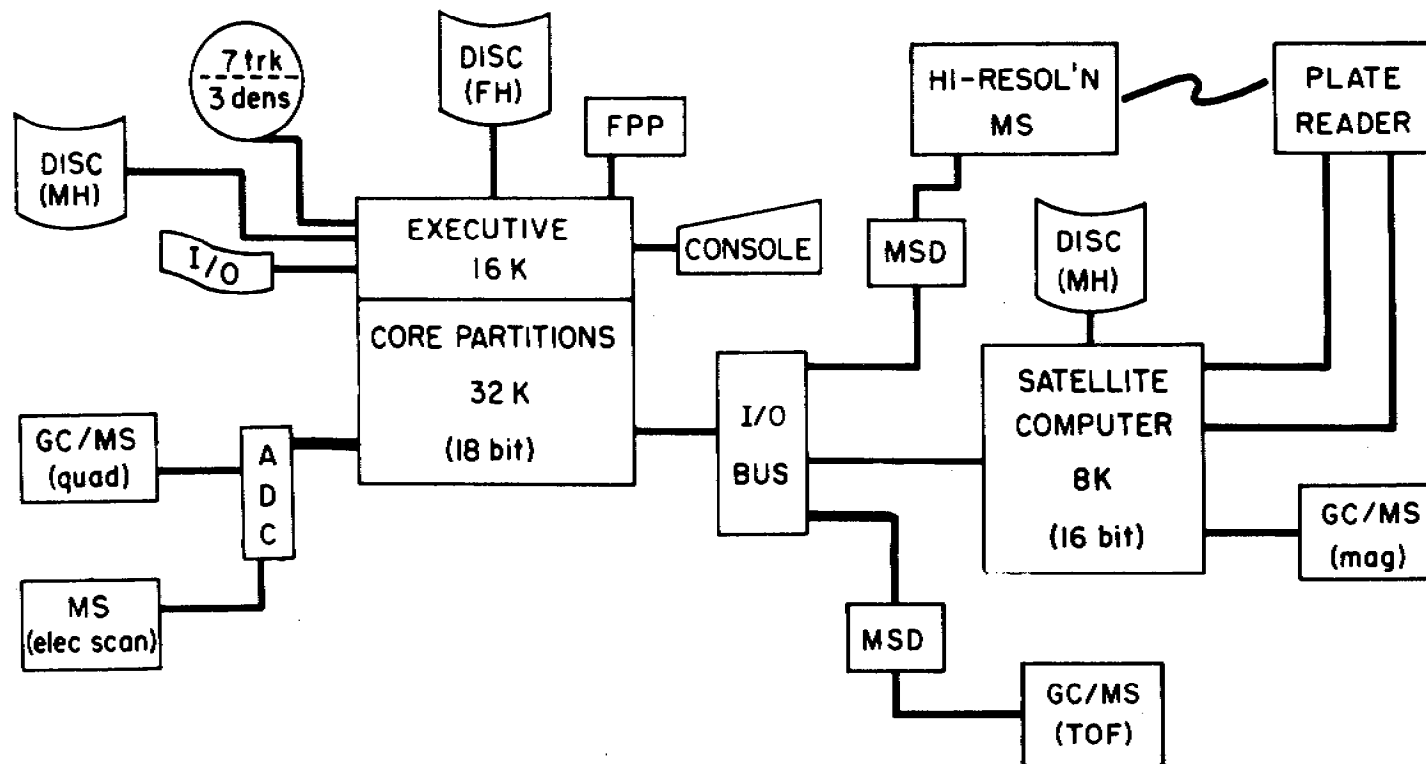


Fig. 2

$$-\eta = \sum_{i=1}^n p_i \log p_i$$

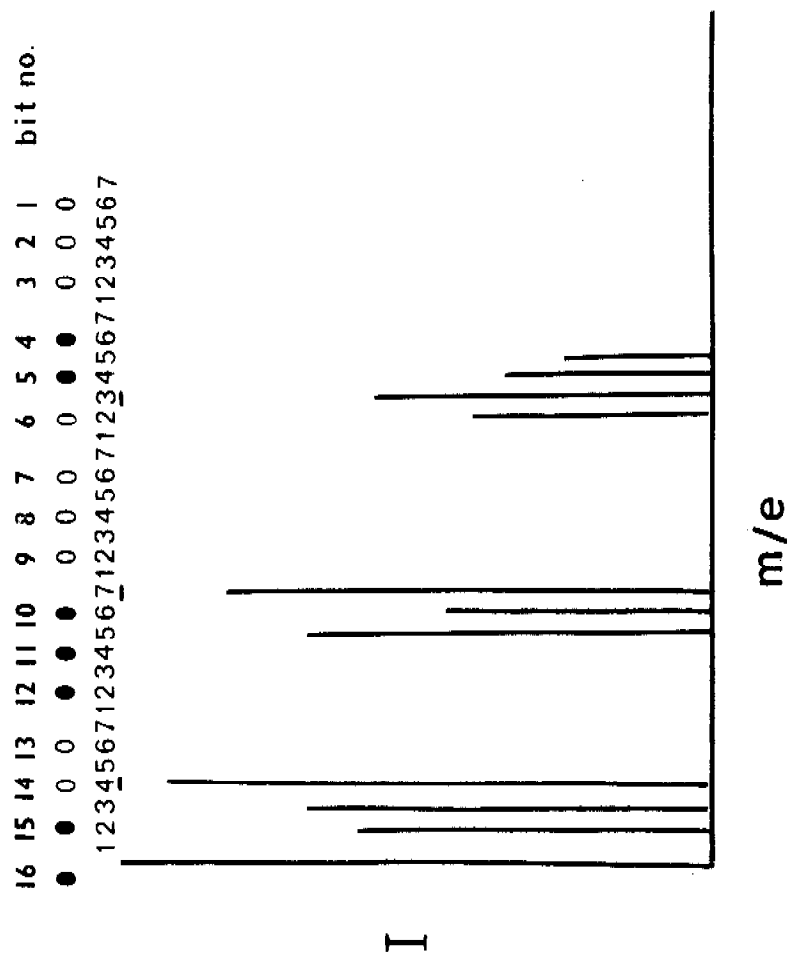
Fig. 3

$$J(1,2) = N_1 \sum_i^n (p_{1i} - p_i) \log_e \frac{p_{1i}}{p_i} + N_2 \sum_i^n (p_{2i} - p_i) \log_e \frac{p_{2i}}{p_i}$$

$$p_i = \frac{p_{1i} + p_{2i}}{2}$$



FIG. 4



References:

1. C. Merritt, Jr., D.H. Robertson, R.A. Graham and T.L. Nichols, Proceedings of the 20th Annual Conference, ASMS, page 355.
2. D.H. Robertson, J. Cavagnaro, J.B. Holz and C. Merritt, Jr., Proceedings of the 20th Annual Conference, ASMS, page 359.
3. D.H. Robertson and C. Merritt, Jr., Proceedings of the 21st Annual Conference, ASMS, page 65.
4. S.L. Grotch, Anal. Chem. 45, 3(1973).

Recursive Digital Filtering Applied to a Mini-Computer  
Data Acquisition System Proposed for Army Wind Tunnels

R. P. Reklis  
U.S. Army Ballistic Research Laboratories  
Aberdeen Proving Ground, Maryland

A recursive digital filtering scheme designed for use with wind tunnel data will be discussed. The filter has been designed in such a fashion that it will not distort low order polynomial data. This filter is to be used in a proposed mini-computer based data acquisition system. This system will automate several functions required in wind tunnel data taking and will gather and display data graphically. Its design will be discussed briefly.

## INTRODUCTION

Wind tunnel data are composed of a series of data sets. Each set is made up of several runs. A run is composed of data taken while a spinning model slows down or while a model is swept through angle of attack. Data are often linear with spin rate or angle of attack and are generally fit with a low order polynomial. At the end of a run calibration is checked and another run is made at an altered angle of attack or roll position. A data set is ended when it is necessary to stop the flow of air in the tunnel in order to make further alterations in the test.

A data system has been proposed that will automate all of the action which takes place during a set of runs. When the set is complete data will be plotted for use by the test engineer. Data will be processed in the following manner. A tape will be made of digitized data in nearly the form they are produced by the analog to digital converter. This tape will be made on line for future analysis on the main BRL computer and for possible playback through the data acquisition system. Data will be simultaneously filtered digitally and stored in core buffers. At the end of each run data will be taken from these core buffers and written on disc. At the end of a series of runs data will be plotted. The system will allow data from any run in the entire series to be plotted. It will be possible to overlay graphs and to alter the scale of plots from the keyboard. The hardware for this mini-computer system is diagrammed in Figure 1. It consists of a Harris 6024/5 CPU which is currently owned together with various peripherals. The Astrodata seen in Figure 1 refers to the data acquisition system currently in use. It is a hard wired system that writes data on magnetic tape. The fifty instrumentation amplifiers, multiplexer wiring, and A/D converter contained

in this system will be carried over. The tunnel control subsystem sets switches which control tunnel operation. All other peripherals are standard.

The Astrodata amplifiers described above include switch selectable Butterworth filters. It is desirable to increase the variety of filtering available and to allow the selection of filtering after test data are collected. This last feature in particular suggests digital filtering. The scheme selected must use a minimum amount of both memory and time. A recursive scheme seems best suited. It is in addition necessary that the filter not distort data that may be fit with a low order polynomial.

This paper begins with a review of some background material. Digital filtering is developed on this background that fits the needs described above. Random noise effects, start up effects, numerical accuracy, and deviation from the limiting continuous behavior are then discussed. Further background material is available from several references. Reference 1 contains reprints of important papers and provides a good review of the subject. The paper by C. M. Rader and B. Gold<sup>2</sup> is of particular interest. A related discussion of polynomial smoothing is given by Blackman<sup>3</sup>.

## SOME USEFUL CONCEPTS

### Filtering Properties of Derivatives

It is, of course, obvious that the derivative of a jittery function is likely to be even more jittery. It should not be surprising, therefore, that the analysis of many filters leads to differential equations in which the derivatives are taken of the filter output, which is to be smoothed to obtain the filter input, which is rough. For example, the differential equation that describes the common resistor-capacitor filter is,

$$C \, d V_{\text{out}}/dt + V_{\text{out}}/R = V_{\text{in}}/R .$$

Such an equation describes a method by which the filter input may be obtained from the output. Clearly it is desirable to invert this equation to a form that gives the filter output from the filter input. Such an inversion leads to the Green's function and to the transfer function of the problem.

### Transfer Function and Green's Function

The differential equations that describe many filters have the form

$$\sum_{n=0}^N a_n \frac{d^n}{dt^n} V_{\text{out}} = V_{\text{in}} , \quad (1)$$

where the  $a_n$ 's are constants and the zeroth derivative is taken to be one.

The operator in Eq. (1) applied to  $V_{\text{out}}$  is linear and for this reason solution pairs  $V_{\text{out}}, V_{\text{in}}$  have the property that  $V_{\text{out}_1} + V_{\text{out}_2}, V_{\text{in}_1} + V_{\text{in}_2}$  is a solution pair if  $V_{\text{out}_1}, V_{\text{in}_1}$ , and  $V_{\text{out}_2}, V_{\text{in}_2}$  are. Thus, Eq. (1) can be rewritten by expressing  $V_{\text{out}}$  and  $V_{\text{in}}$  in terms of some complete set of functions and solving for each component. This is usually done by

expressing  $V_{out}$  and  $V_{in}$  as Fourier transforms since this procedure diagonalizes the problem, that is,

$$\left[ \sum_{n=0}^N a_n \frac{d^n}{dt^n} \right] e^{i\omega t} = \left[ \sum_{n=0}^N a_n (i\omega)^n \right] e^{i\omega t}$$

in which  $\sum_{n=0}^N a_n (i\omega)^n$  is the eigenvalue for the eigen function  $e^{i\omega t}$ .

The expression,

$$1 / \sum_{n=0}^N a_n (i\omega)^n ,$$

is known as the transfer function since if a signal  $e^{i\omega t}$  is input to the filter the output will be,

$$e^{i\omega t} / \sum_{n=0}^N a_n (i\omega)^n ,$$

and, thus, Eq. (1) has been inverted in frequency space.

A return from frequency to time space can be made by carrying through the inverse Fourier transforms. Defining the Fourier transform pair as:

$$F(t) = 1/\sqrt{2\pi} \int_{-\infty}^{\infty} f(\omega) e^{i\omega t} d\omega \quad (2)$$

$$f(\omega) = 1/\sqrt{2\pi} \int_{-\infty}^{\infty} F(t') e^{-i\omega t'} dt' \quad (3)$$

gives,

$$V_{out}(t) = (1/2\pi) \int_{-\infty}^{\infty} dt' V_{in}(t') \left\{ \int_{-\infty}^{\infty} d\omega \left[ e^{i\omega(t-t')} / \sum_{n=0}^N a_n (i\omega)^n \right] \right\}.$$

This expression is of the form

$$V_{out}(t) = \int_{-\infty}^{\infty} dt' V_{in}(t') g(t - t'),$$

where the Green's function  $g(t - t')$  is given by

$$g(t - t') = (1/2\pi) \int_{-\infty}^{\infty} d\omega \left[ e^{i\omega(t-t')} / \sum_{n=0}^N a_n (i\omega)^n \right]. \quad (4)$$

The integral in Eq. (4) can be readily evaluated by use of the residue theorem if the zeros of the polynomial,

$$P_N = \sum_{n=0}^N a_n x^n, \quad (5)$$

are known.

For a physically realizable filter the Green's function must be zero for  $t < t'$  as the filter cannot exhibit a response to events in the future. It is worthwhile noting that this will only be the case if the polynomial in Eq. (5) has no zeros in the right half plane.

#### The Aliasing Problem

A digital filtering scheme acts on a finite number of discretely spaced data points to which the continuous analysis developed above does not



directly apply and it is necessary to carry over the various equations to the discrete case. Equation (1) for example can be rewritten in terms of finite differences or, in general, as,

$$\sum_{m=0}^N a_m H(n-m) = D(n), \quad (6)$$

where  $H$  is the output of the filter,  $D$  is the data set and the  $a_m$ 's are constants. Such an equation is linear and Fourier analysis applies. The transform pair becomes,

$$D(n) = (1/\sqrt{2M+1}) \sum_{k=-M}^M d(k) \exp [2\pi i k n / (2M+1)], \quad (7)$$

$$d(k) = (1/\sqrt{2M+1}) \sum_{n'=-M}^M D(n') \exp [-2\pi i k n' / (2M+1)], \quad (8)$$

for  $2M+1$  data points.

This form should be very familiar to anyone with solid state experience; a crystal lattice is a discrete space and similar forms are used. Just as the finite nature of the lattice folds momentum space in upon itself giving rise to the Brillouin zone, frequency space is folded in the digital filter problem. For example, there is no way to distinguish sixty Hz noise from a D.C. signal if data are sampled at sixty Hz (see Figure 2) as sixty Hz is folded into DC at this sample rate. This phenomenon is known as aliasing and implies that a data rate must be selected with care.

The relationship between the continuous and discrete Fourier transforms is obtained by sampling a continuous function whose transform is known and

calculating the discrete transform Eq. (9) , by combining Eq. (2) with Eq. (8).

$$d(k) = (1/\sqrt{2M+1}) \sum_{n=-M}^M \left\{ \left[ (1/\sqrt{2\pi}) \int_{-\infty}^{\infty} f(\omega) e^{i\omega n\tau} d\omega \right] \exp [2\pi i k n / (2M+1)] \right\} , \quad (9)$$

where  $\tau$  is the sampling period. This equation simplifies to:

$$d(k) = (1/\sqrt{2\pi(2M+1)}) \int_{-\infty}^{\infty} d\omega \left\{ f(\omega) \sin [(2M+1)(\omega\tau - 2\pi k / (2M+1)) / 2] / \right. \\ \left. \sin [(\omega\tau - 2\pi k / (2M+1)) / 2] \right\} . \quad (10)$$

The implications of Eq. (10) are that the discrete coefficient samples the continuous coefficients in a window of half width  $2\pi/(2M+1)$  about  $\omega = 2\pi k / \tau(2M+1)$ . A word of caution is urged in this interpretation, however. A noisy signal with components at frequencies much higher than the data sampling rate will produce an essentially random data component. Clearly, if the  $D(n)$  in Eq. (8) are random the Fourier components  $d(k)$  will also be random. The problem of obtaining useful numerical frequency spectra is discussed in reference 4. As long as the Green's function is broad enough to include many data points and as long as the band pass is much wider than  $2\pi/(2M+1)$  statistical effects will average out. This is demonstrated below.

#### Linearity

These filters are designed for use with data that approximates a low order polynomial in addition noise at particular frequencies may be present as well as a random variation in the data points. Finally, the solution to the equations that describe these filters is an initial value problem and

the response depends on starting conditions. Fortunately, these equations are linear and the response to each of these signal components may be treated separately.

### SHIFT FILTERS

The result of applying the operator  $O$ ,

$$O = \left[ \sum_{n=0}^N (\alpha^n/n!) d^n/dt^n \right], \quad (11)$$

to a polynomial in  $t$  of order  $N$  is to shift the polynomial, without distortion, by an amount  $\alpha$  along the  $t$  axis; that is,

$$O \left[ \sum_{m=0}^N c_m t^m \right] = \sum_{m=0}^N c_m (t + \alpha)^m,$$

where  $\alpha$  and the  $c_m$  are constants.

The transfer function for this operator is,

$$1 / \left[ \sum_{n=0}^N (i\alpha\omega)^n/n! \right], \quad (12)$$

which for small  $\alpha\omega$  is just  $e^{-i\alpha\omega}$ , and has magnitude one. The band pass for a filter described by Eq. (11) will, therefore, be flat for small  $\omega$ . For large  $\omega$  only the last term in the polynomial in Eq. 12 need be kept and the filter will cut out high frequency as  $N!/(\alpha\omega)^N$ . A filter described by this operator will have the desirable property of a flat band pass and will shift without distortion a data signal that approximates a low order polynomial. Unfortunately, only filters  $N = 1, 4$  are realizable as the polynomial,

$$P_N(X) = \sum_{n=0}^N X^n/n!, \quad (13)$$

has zeros in the right half plane for  $N > 4$ .

The band pass curves of the first four filters in this series are shown in Figure 3. These have been adjusted, by proper choice of  $\alpha$ , so that they have the same 3 db. cut off frequency. The relationship between  $\alpha$  and the 3 db. cut off frequency is given in Table I.

The Green's functions of the first four filters can be obtained from the residue theorem and Eq. 4 as stated above and can be shown to be,

$$g(t - t') = \begin{cases} - (N!/\alpha) \sum_{n=1}^N \exp [Z_n (t - t')/\alpha] / Z_n^N, & t \geq t', \\ 0, & t < t', \end{cases} \quad (14)$$

where the  $Z_n$ 's are the zeros of the polynomial, Eq. 13. These zeros are plotted in Figure 4.

Note should be made that the filters for  $N = 1$  and 2 are Butterworth filters; however, the filters for  $N = 3$  and 4 are not.

#### NUMERICAL SIMULATION

The desirable feature of this series of filters is that low order polynomial data is not distorted. This is the feature that led to the development of the operator in Eq. 11 and it can be used to simulate the operator numerically. Rearranging Eq. 6 to solve for the last output of the filter  $H(n)$  in terms of the previous outputs and the current datum point  $D(n)$  gives,

$$H(n) = A_0 D(n) + \sum_{m=1}^N A_m H(n-m). \quad (15)$$

This is the useful form of the filter algorithm and gives the current filter output in terms of past filter outputs and current data. The A's can be solved for by insisting that,

$$D(n) = H(n+a) , \quad (16)$$

when

$$H(n) = \sum_{\ell=0}^N a_{\ell} n^{\ell} . \quad (17)$$

Equations 16 and 17 imply that the filter algorithm given by Eq. 15 will have the appropriate shifting property for polynomial data. Applying Eq. 16 and Eq. 17 to Eq. 15 leads to the set of equations,

$$1 = A_0 (1+a)^{\ell} + \sum_{m=1}^N A_m (1-m)^{\ell} , \quad (18)$$

$$\ell = 0, 1, \dots, N,$$

which can be solved for the  $A_m$ 's giving,

$$A_0 = N! / [(1+a)(2+a)\dots(N+a)] , \quad (19)$$

$$A_m = - [a/(m+a)] (-1)^m N! / [m! (N-m)!] . \quad (20)$$

The connection between the discrete operator, Eq. 15, and the continuous operator, Eq. 11, can be seen by writing Eq. 15 with the coefficients given in Eq. 19 and Eq. 20 in terms of difference operators as,

$$D(n) = H(n-N) + \sum_{\ell=1}^N \left[ \frac{(N+a)(N-1+a)\dots(N-\ell+1+a)}{\ell!} \nabla^{\ell} H(n-N+\ell) \right] ,$$

where

$$\nabla^{\ell} H(n) = \sum_{k=0}^{\ell} \binom{\ell}{k} (-1)^k H(n-k).$$

For large  $a$  this becomes,

$$D(n) = \sum_{\ell=0}^N [(a\tau)^{\ell}/\ell!] \nabla^{\ell} H(n-N+\ell)/\tau^{\ell}.$$

Let  $H$  sample the function  $F(t)$  at increments of  $\tau$  and fix  $a\tau = \alpha$  a constant, then by definition the right hand side of the above equation becomes,

$$\sum_{\ell=0}^N (\alpha^{\ell}/\ell!) d^{\ell} F(t)/dt^{\ell},$$

as  $\tau$  approaches zero.

Thus, the discrete filter given in Eq. 15 with coefficients given by Eq. 19 and Eq. 20 is equivalent to the continuous operator given in Eq. 11 in the limit of small sampling period with  $\alpha = a\tau$ .

Such equivalence is often shown by fitting the discrete data points with a continuous function of some specific form that eases the translation from continuous to discrete equations maintaining the band pass characteristics of the filter<sup>5</sup>. This was not done in translating the shift filter, since the property that remains constant in this translation is the shift property and not the band pass.

#### RANDOM NOISE

The average response,  $\overline{H(n)}$ , of the filter given in Eq. 15 to random data of distribution,

$$e^{-(D/\sigma)^2} / \sigma \sqrt{\pi} ,$$

is zero. The average squared response gives a measure of the randomness of the filtered data and is given by,

$$\overline{H(n)^2} = \overline{D(n)^2} \left[ \sum_{\ell=0}^{\infty} g^2(n-\ell) \right] ,$$

where  $g(n-\ell)$  is the discrete Green's function, i.e. the response  $H(n)$  to  $D(n-\ell) = 1$  and  $D$  otherwise zero. The random noise attenuation factor,

$$\sum_{\ell=0}^{\infty} g^2(n-\ell) ,$$

can be calculated in the large  $a$  limit from the continuous Green's functions and becomes,

$$= \left[ (N!)^2 / a \right] \sum_{k=1}^N \sum_{kk=1}^N \left[ 1/Z_k^N Z_{kk}^N (Z_k + Z_{kk}) \right] . \quad (21)$$

The value of these factors are given in Table II while Figure 5 shows a plot of the attenuation factors for small  $a$ .

#### START UP

As mentioned previously the filter output depends on starting conditions. As may be seen in Eq. 14 the magnitude of the Green's functions damps out exponentially and consequently starting conditions may be ignored after a certain startup time that goes as  $a$  divided by the real part of the zero of the polynomial in Eq. 13 that is closest to the imaginary axis. These damping factors are given in Table III. Clearly, some improvement may be

made if some scheme can be used to start the filter algorithm by assigning  $H(1-m)$ ,  $M = 1, 2 \dots N$  some appropriate values. This may be done by setting all  $H(1-m)$  equal to  $D(1)$  or by curve fitting the first few data points and extrapolating backwards, etc.

#### NUMERICAL ACCURACY

The response of the digital shift filters to a signal  $D(n) = 1$  may be calculated from Eq. 15 as,

$$H(n) = A_0 + \sum_{m=1}^N A_m, \quad (22)$$

which, by Eq. 18, is  $H(n) = 1$ . Clearly, the sum in Eq. 22 represents the D.C. gain of the filter. It can be shown that the values,

$$G_{\ell,k} = A_0 a^{\ell-k} + \sum_{m=1}^N (-m)^{\ell-k} A_m \ell!/k! (\ell-k)!, \quad k \leq \ell,$$

give a measure of the distortion in the form of an  $n^k$  term present in the filter output when  $n^\ell$  is input. Note that the  $G_{\ell,\ell}$ 's reduce to the gain calculated from Eq. 22. The calculation of these distortion terms gives a measure of the numerical accuracy to be expected from the algorithm.

#### SMALL SHIFT BEHAVIOR

Due to the aliasing problem the band passes of digital filters are periodic in  $\omega$  with period  $2\pi/\tau$ . It is, therefore, necessary to use them in conjunction with some electronic filtering to insure that signal components of frequency  $\omega > \pi/\tau$  are minimal. The data acquisition system



used with the wind tunnel application uses a second order Butterworth filter with a cut off frequency set at  $\omega \approx \pi/\tau$ . A choice of  $a \geq 1$  will then insure that final filter output is predominantly controlled by the digital filter. The question then arises as to whether the large  $a$  approximation is appropriate for  $a \geq 1$ . The random noise attenuation factor being the integral of the square of the Green's function serves as a good single parameter for a study of the closeness to which the behavior of the filter is approximated by the large  $a$  formulas. As can be seen from Figure 5 differences can be expected and the small  $a$  band passes and Green's functions must be calculated.

The band pass may be obtained from the numerical transfer function,

$$\left[ 1 - \sum_{m=1}^N A_m \exp(-i\omega\tau m) \right] / A_0 ,$$

$$\omega = 2 \pi i k / \tau (2M+1), k = -m, \dots, m$$

The attenuation curves are plotted in Figure 6 for  $a = 1$  and in Figure 7 for  $a = 5$ . Figure 8 shows the 3 db. roll off frequency as a function of  $\omega$ ; it is multiplied by  $a$  to show the approach to the large  $a$  limit.

The small  $a$  Green's functions are easily obtained by applying a pulse to the filter, i.e.  $D(1) = 1$ ,  $D(n) = 0$ ,  $n \neq 1$ . The variable of interest when applying the filters is the width of this Green's function since this determines the time necessary for errors in the initial conditions to damp out. These widths are plotted in Figure 9. The criterion used in calculating them is that integral of the square of the Green's function has reached ninety percent of its limiting value. For practical purposes the width of

the Green's function is about twice the value calculated by this criterion, the factor of two arising from the fact that the Green's function was squared.

The 3 db. roll off frequency and random noise factors for small  $a$  are given in Tables IV and V.

### CONCLUSION

The filter algorithms discussed in this paper have been linear in nature. That is the action of the filters on various data components may be analyzed separately. As has been shown the main data signal will be preserved without distortion if it is in the form of a small order polynomial. The attenuation of noise at particular frequencies has been discussed. In general, the band pass becomes wider with increasing filter order and a particularly noticeable spike or resonance is observed in the fourth order filter which may pose a problem if noise is present at this frequency. The fourth order filter will also pass more random noise than the others, and it is also true that the fourth order filter will require a greater amount of time to damp out starting effects or noise spikes. All four filter types are useful if consideration is taken of the peculiarities found in order four, however. Figure 10 shows the effects of all four orders on a sample of wind tunnel data containing a large 60 Hz noise component. Note that this component appears at 20 Hz due to the 40 Hz sampling rate and is eliminated by the filtering.

### ACKNOWLEDGMENTS

The author would like to acknowledge the help of Mr. Michael Conboy of the University of Massachusetts in finding references.

Table I

Attenuation Factors and 3 db. Cut Off Frequencies for Shift Filters

<u>Order</u>	<u>Attenuation Factor</u>	<u>3 db. Frequency (<math>H_z</math>)</u>
1	$1/\sqrt{1 + (\omega\alpha)^2}$	$f_3 = .276/\alpha$
2	$1/\sqrt{1 + (\omega\alpha)^4/4}$	$.296/\alpha$
3	$1/\sqrt{1 - (\omega\alpha)^4/12 + (\omega\alpha)^6/36}$	$.390/\alpha$
4	$1/\sqrt{1 - (\omega\alpha)^6/72 + (\omega\alpha)^8/576}$	$.499/\alpha$

Table II

Random Noise Attenuation Factors for Large  $a$  Filters

<u>Filter Order</u>	<u>Attenuation Factor</u>
1	.5 /a
2	.5 /a
3	.75/a
4	1.5 /a

Table III

Green's Function Damping Factor

<u>N</u>	<u>Factor</u>
1	- 1
2	- 1
3	- 1.42
4	- 3.69

Table IV

Band Edge Frequencies (Hz) for Small a Filters (3 db)

Data Rate = 40 Hz for (120 Hz multiply by 3)

<u>a</u>	Filter Order			
	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>
1	8.40	6.68	7.07	7.64
2	4.62	4.14	4.71	5.35
3	3.23	3.03	3.59	4.18
4	2.48	2.40	2.90	3.42
5	2.02	2.00	2.43	2.92
6	1.71	1.71	2.10	2.54
8	1.31	1.32	1.66	2.01
10	1.05	1.08	1.36	1.67
12	0.89	0.91	1.16	1.43
14	0.76	0.79	1.01	1.25
16	0.67	0.70	0.89	1.11
18	0.60	0.63	0.80	1.00
20	0.54	0.56	0.73	0.91

Table V  
Random Noise Attenuation for Small  $a$  Filters

	Filter Order			
<u>a</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>
1	.33	.29	.33	.47
2	.20	.18	.22	.36
3	.14	.13	.17	.29
4	.11	.10	.14	.24
5	.091	.087	.12	.21
6	.077	.072	.10	.19
8	.059	.056	.079	.15
10	.048	.046	.066	.12

#### FOOTNOTES

1. Lawrence R. Rabiner and Charles M. Rader, Digital Signal Processing, (IEEE Press, N.Y., 1972).
2. Charles M. Rader and Bernard Gold, Proc. IEEE, 55, 149 (1967).
3. R. B. Blackman, Linear Data-Smoothing and Prediction in Theory and Practice, (Addison-Wesley, Reading, Mass., 1965).
4. J. W. Cooley, A. W. Lewis, and P. D. Welch, The Fast Fourier Transform Algorithm and Its Applications, (IBM, Yorktown Heights, N.Y., 1967).
5. K. Steiglitz, Inform. Contr. 8, 455 (1965).

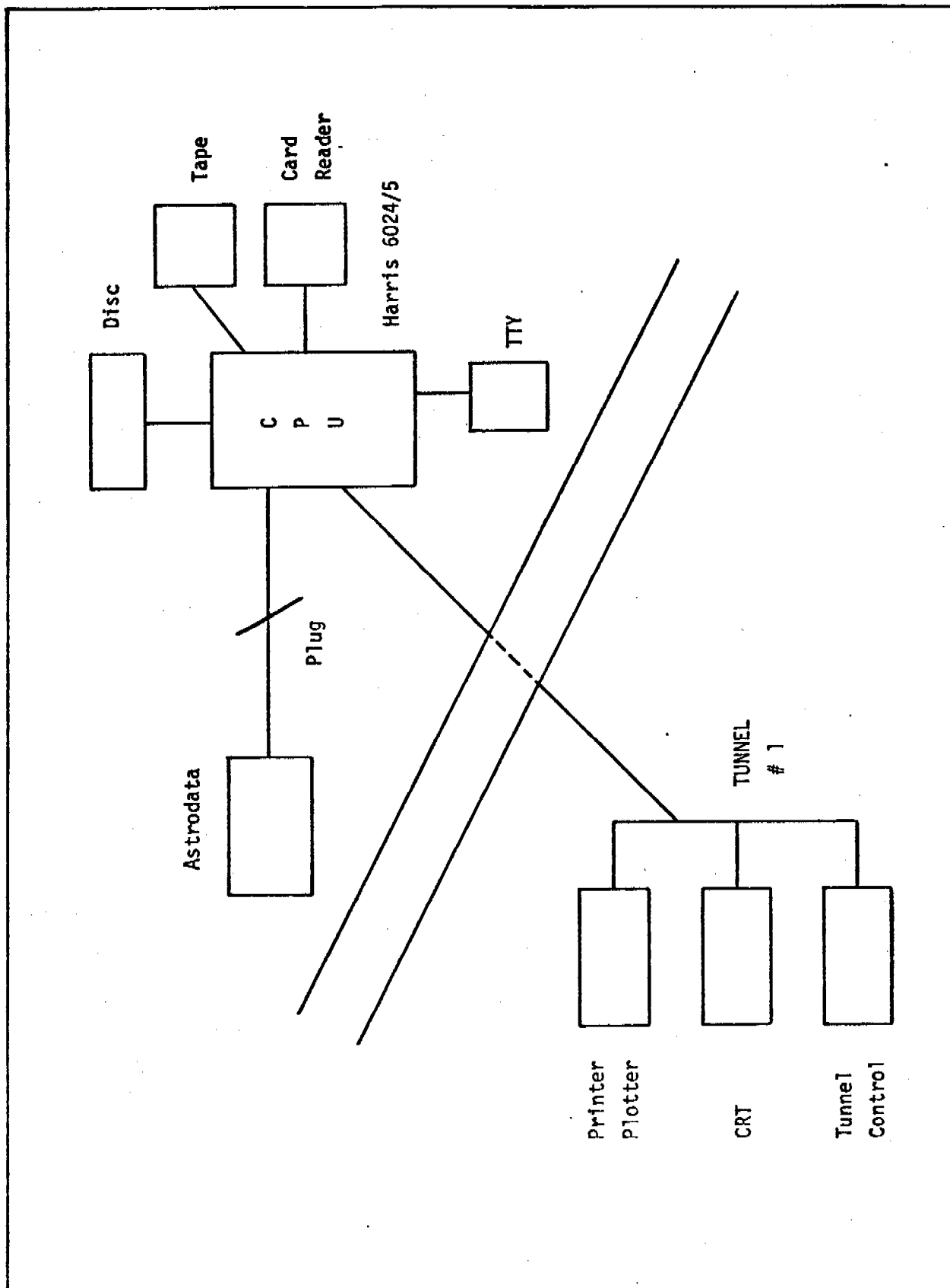


Figure 1. Block Diagram of Data Acquisition and Tunnel Automation System

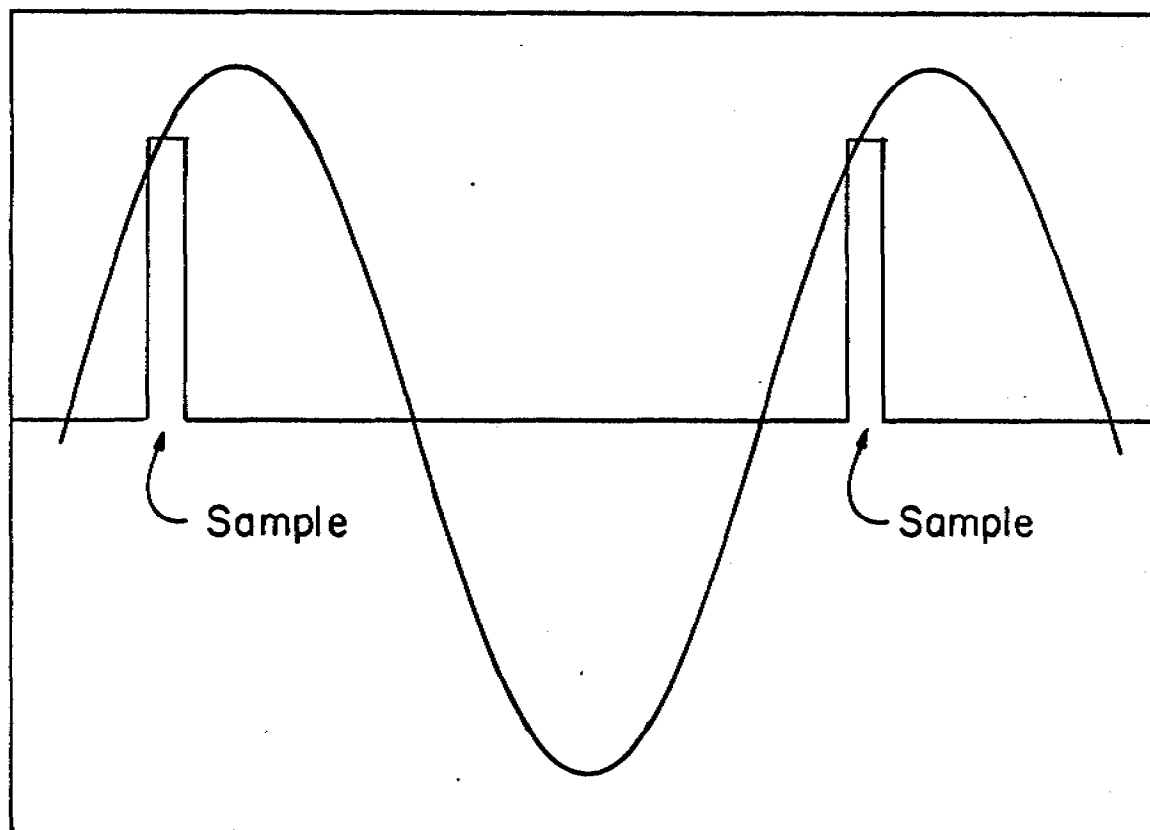


Figure 2. A Sine Wave of Frequency  $1/\tau$  Sampled at  $1/\tau$ . Note that the result may be interpreted as a D.C. signal.

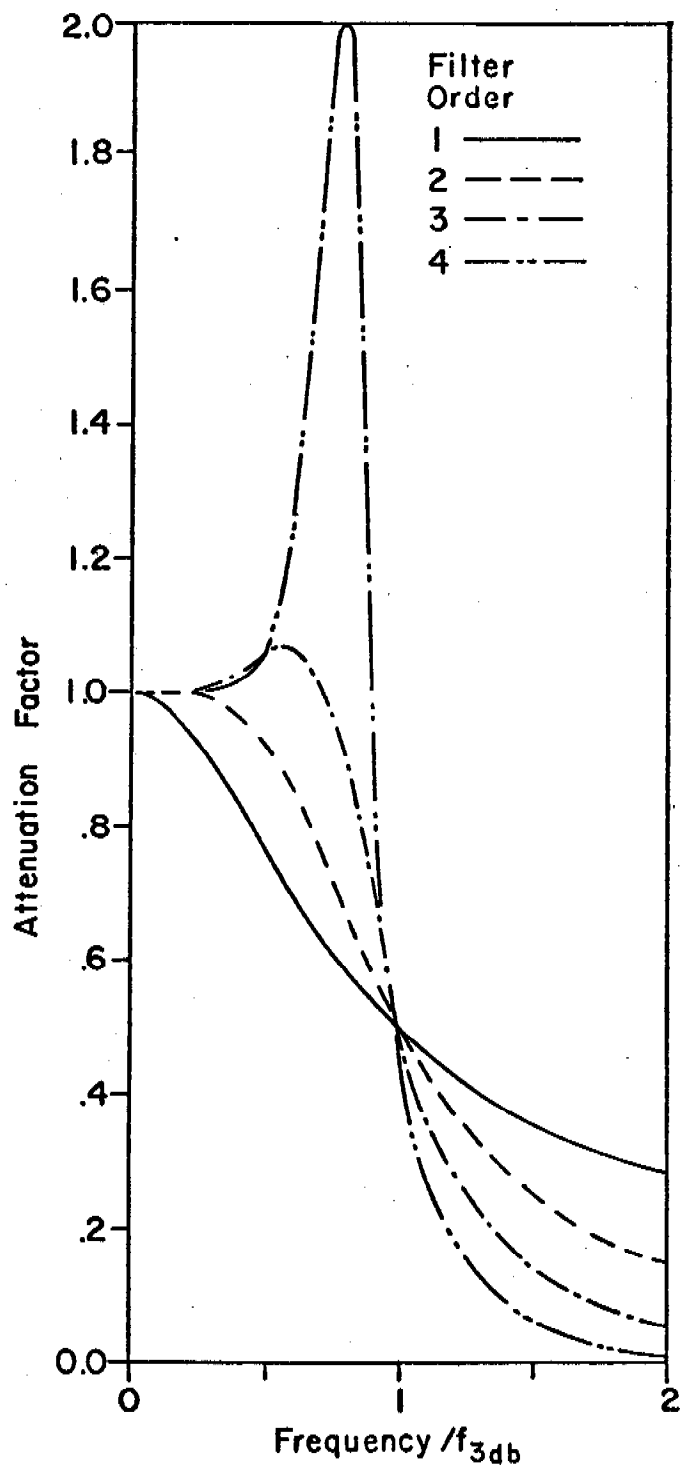


Figure 3. Attenuation Curves Adjusted to Have the Same Band Pass



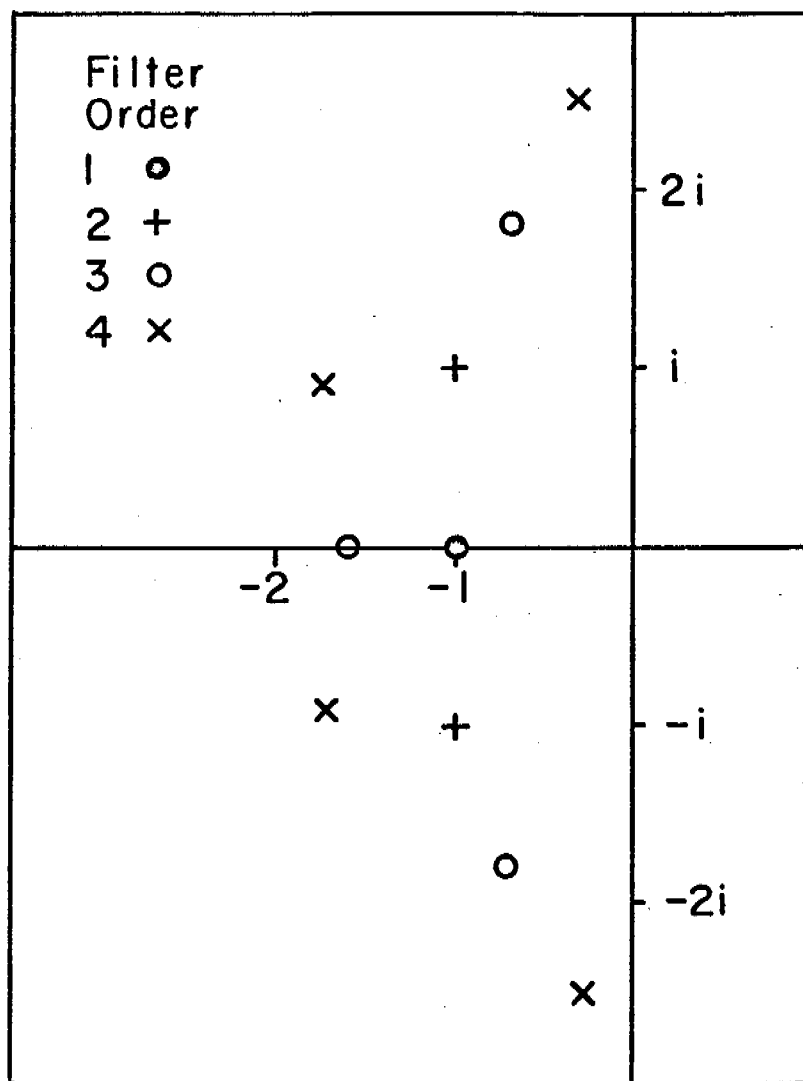


Figure 4. Zeros of  $\sum_{m=1}^N x^m/m!$

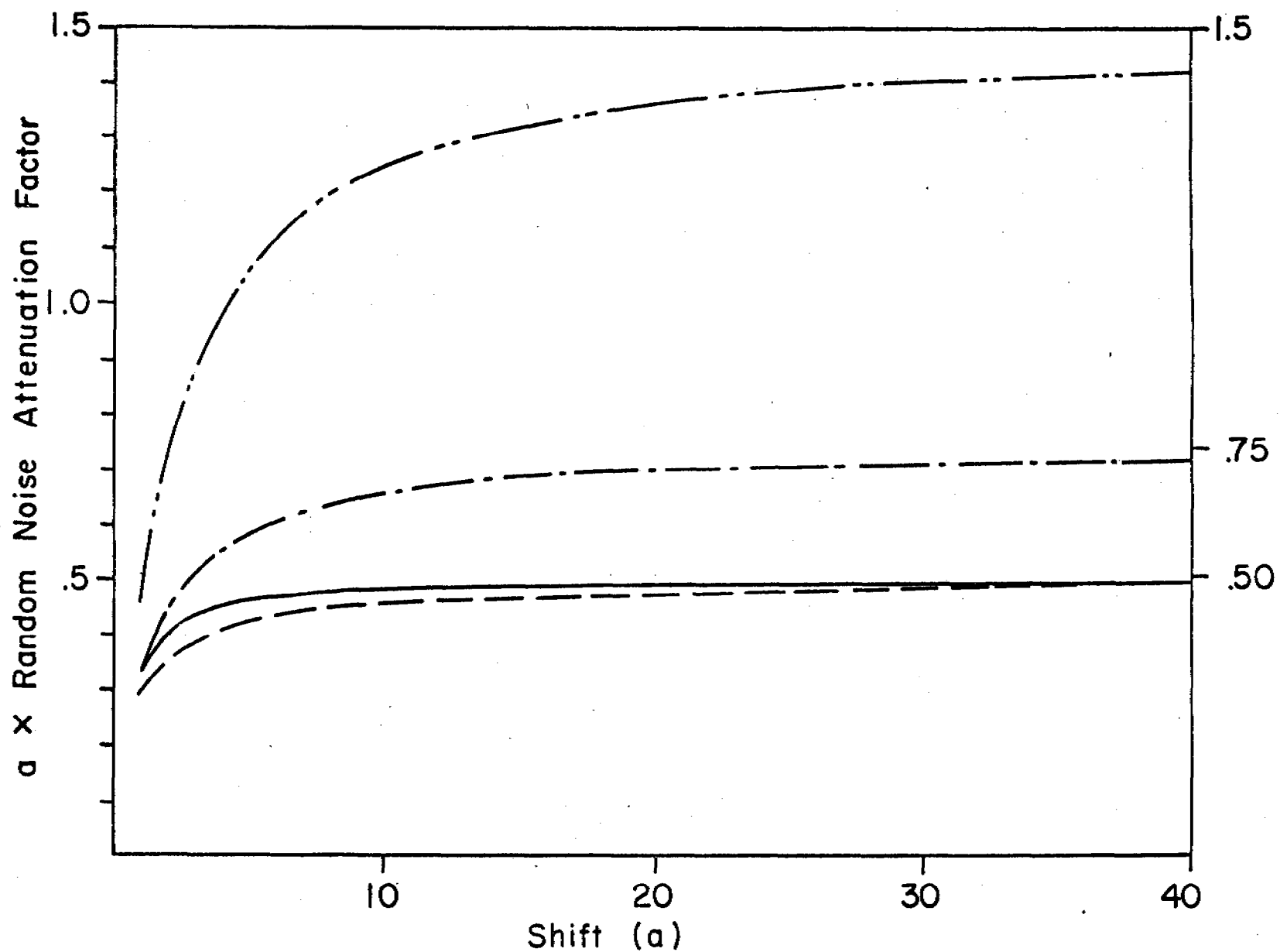


Figure 5. Random Noise Attenuation Factors Multiplied by  $a$  to Show the Approach to the Large  $a$  Asymptote

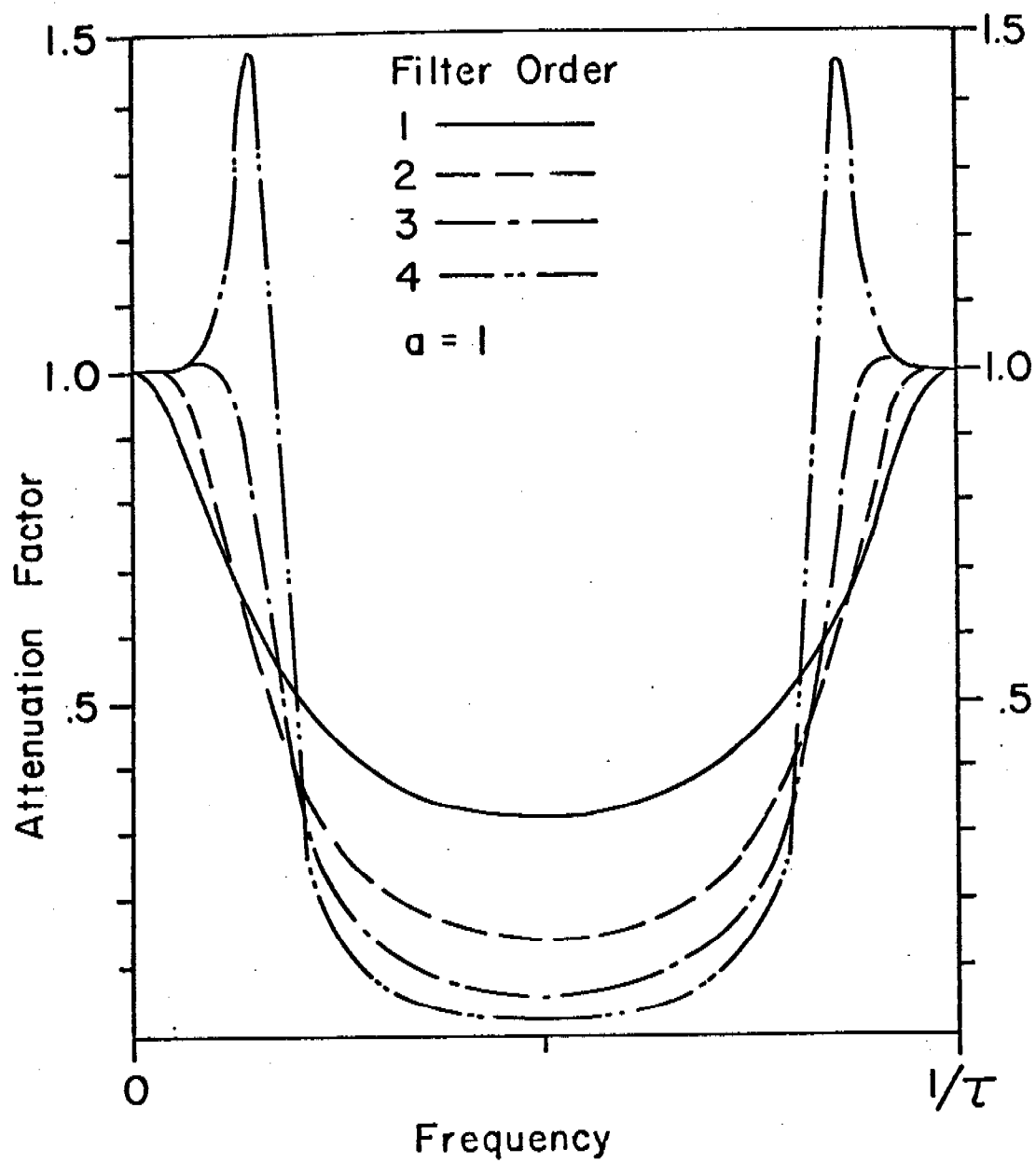


Figure 6. Attenuation Curves for  $a = 1$

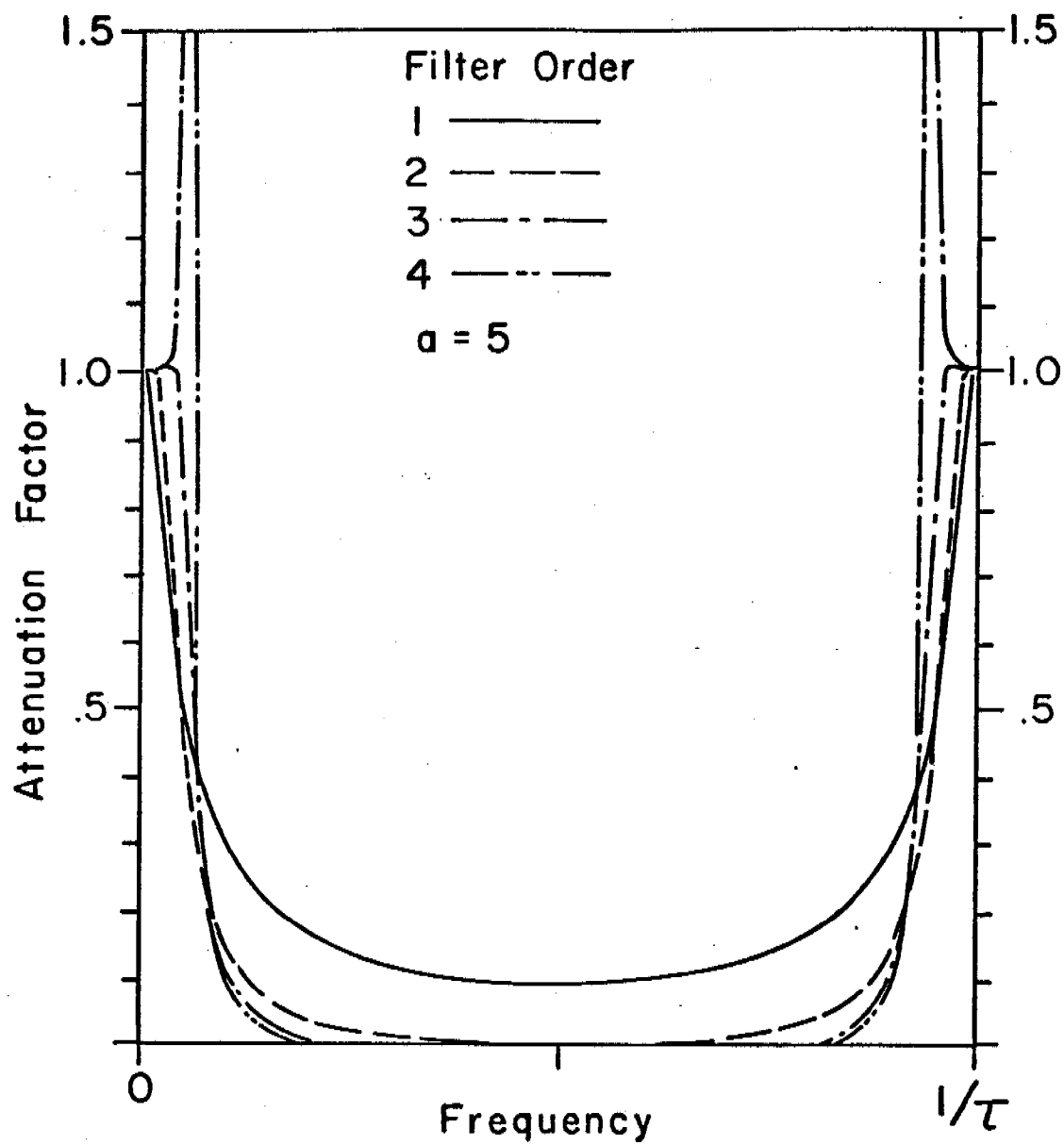


Figure 7. Attenuation Curves for  $a = 5$

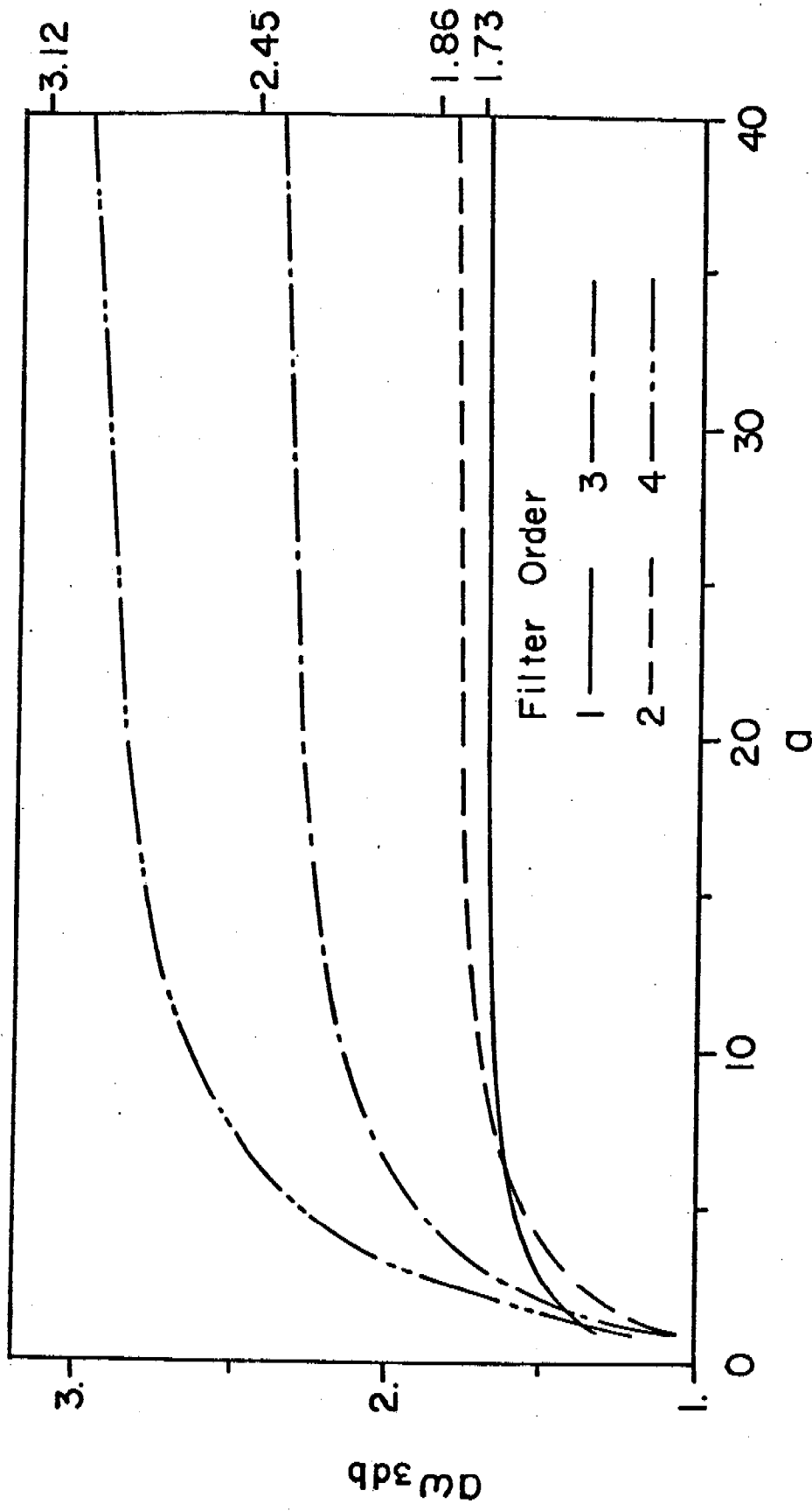


Figure 8. The 3 db. Attenuation Frequency ( $\omega$ ) Multiplied by a to Show Asymptotic Behavior as a Function of a

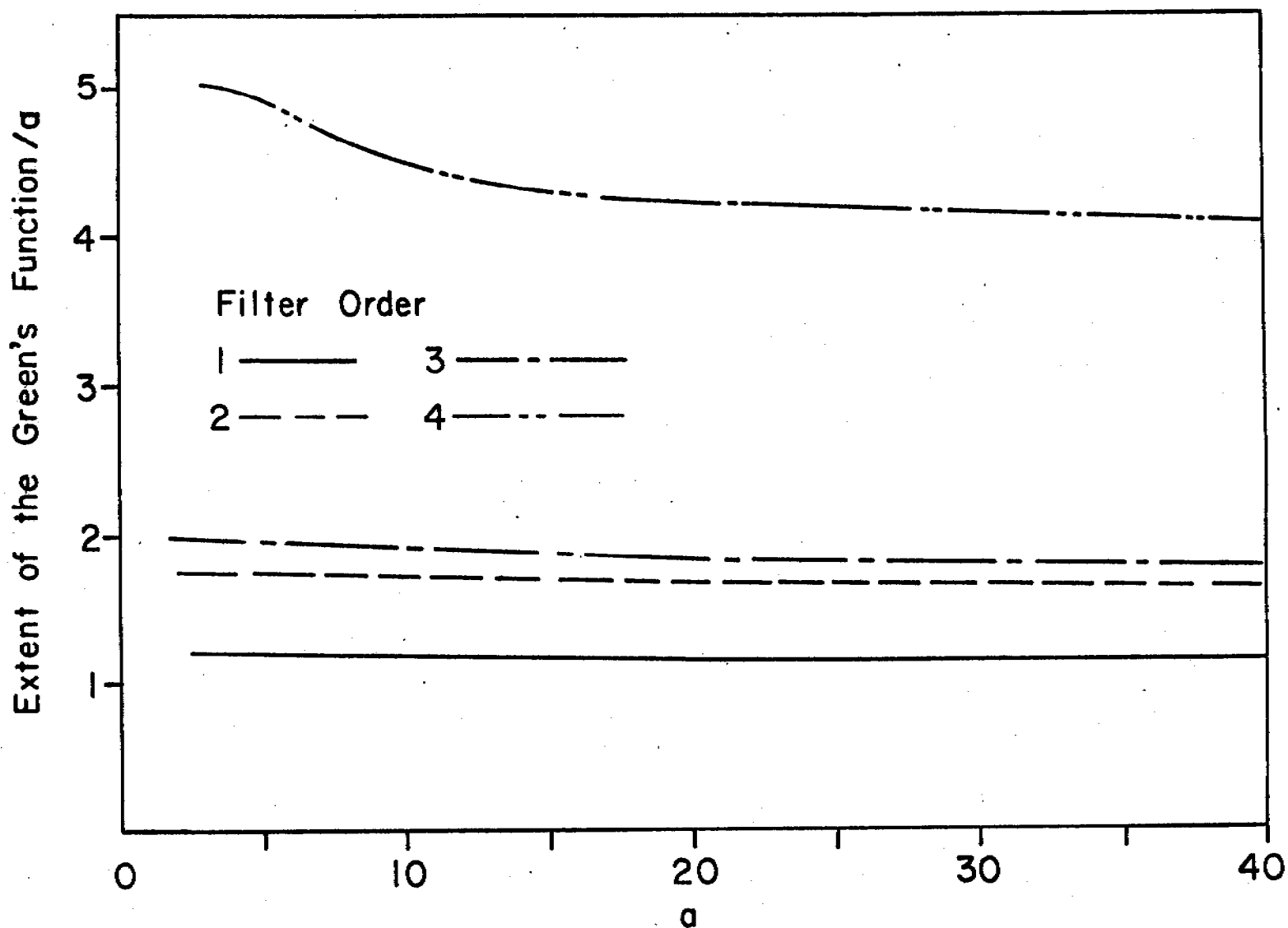


Figure 9. Width of the Small  $a$  Green's Functions Divided by  $a$  to Show the Approach to the Large  $a$  Asymptote. The criterion used to determine width is that the integral at the square of the Green's function has obtained ninety percent of its limiting value.

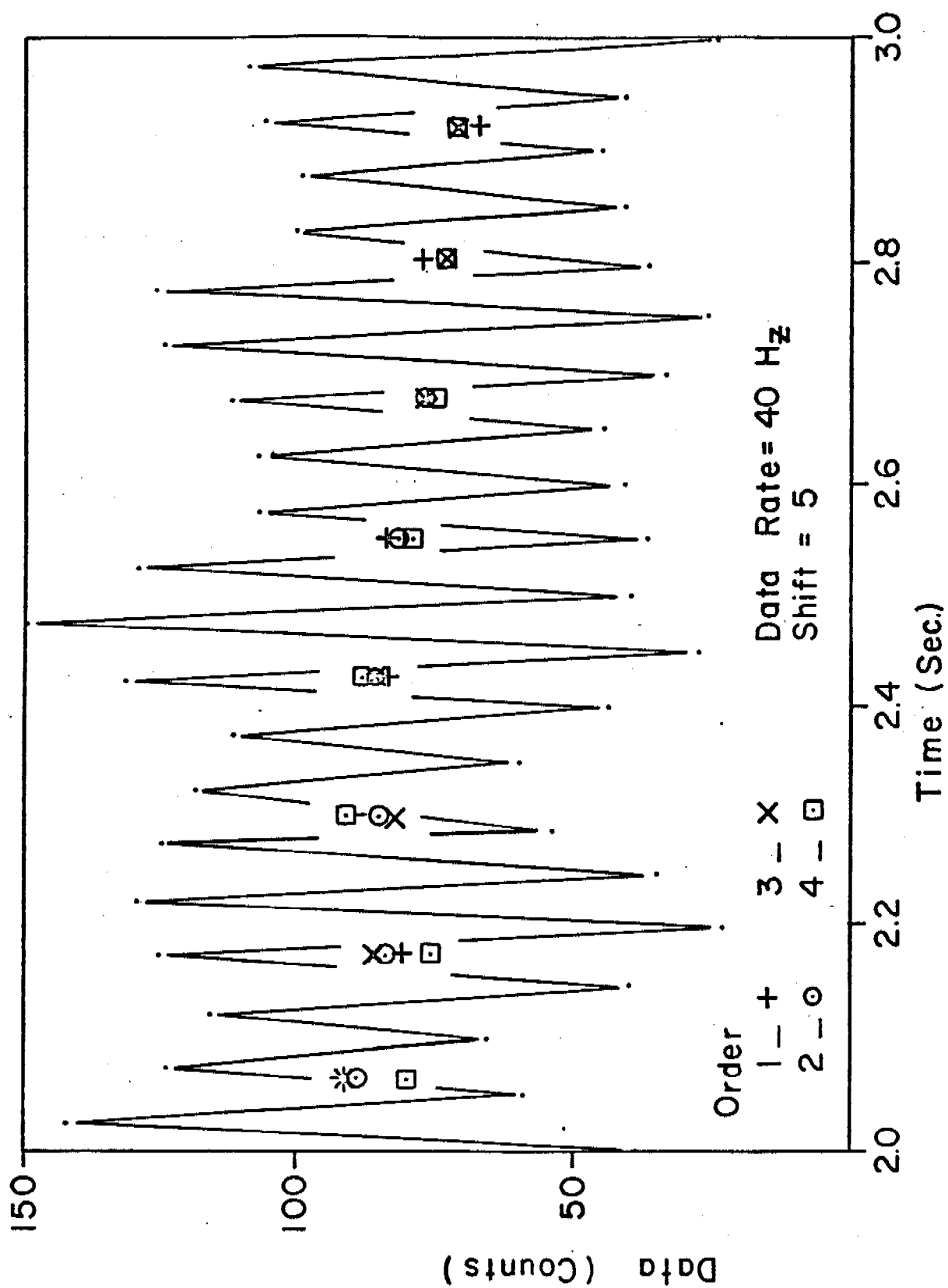


Figure 10. Wind Tunnel Data Showing the Effects of Digital Filtering





# FINITE DIFFERENCE SCHEMES FOR SIMULATING FLOW IN AN INLET-WETLANDS SYSTEM

H. Lee Butler  
U. S. Army Waterways Experiment Station  
Hydraulics Laboratory, Wave Dynamics Division  
Vicksburg, Mississippi 39180

Donald C. Raney  
AME Department  
University of Alabama  
University, Alabama 35486

ABSTRACT. Two numerical models for simulating the tidal hydrodynamics of an inlet, bay or harbor are compared. An implicit finite difference solution to the basic equations of hydrodynamic flow has been developed at the Waterways Experiment Station. This implicit model is compared with an explicit formulation as applied by Masch, 1973. Both models calculate depth averaged velocity components and tidal elevations as a function of position and time during a specified tidal cycle. In addition to the actual bathymetry, the two models include variable bottom roughness, non-linear advective terms in the momentum equations, treatment of regions which are inundated during a portion of the tidal cycle, exposed and submerged barriers, wind stress, and other physical features of the region to be modeled. A discussion of the mathematical formulation and associated finite difference approximations is included. The comparison consists in applying both models to Masonboro Inlet, North Carolina, with identical bathymetric data, boundary conditions, and spatial step size. The hydrodynamic solutions obtained are compared as well as the economics associated with the two models. While, in general, the solutions obtained from the two schemes are comparable, the explicit solution has a considerably more stringent stability criterion limiting the time step. Remedial actions required to overcome stability problems inherent in implicit schemes are discussed.

1. INTRODUCTION. Most numerical simulations of tidal hydrodynamics associated with inlets and bays have been performed using

finite difference schemes based upon an explicit formulation. A code employing such a scheme was developed by Reid & Bodine (1968) and used extensively in treating problems which include possible flooding of low-lying areas and subgrid topographic features. An extension of this work was carried out by Masch, Brandes, and Reagan (1973) for the U. S. Army Coastal Engineering Research Center to help evaluate the degree to which mathematical models can be used to predict the tidal hydrodynamics (exclusive of sediment transport) of an inlet system.

The Wave Dynamics Division, WES Hydraulics Laboratory, has applied implicit finite difference schemes to a variety of problems such as tsunami propagation, simplified tidal models, landslide generated water waves and storm surge calculations (Butler and Durham, 1975). Implicit schemes have been applied successfully by Leendertse (1970, 1971) to regions which include areas that are inundated only during a portion of the tidal cycle. The present work is an extension of the ideas expressed by Leendertse but differs in that the basic equations are written in terms of vertically integrated flows per unit of width rather than velocity, and subgrid features, such as exposed, submerged, and overtopping barriers, are treated.

The principal reason for using the implicit formulation is economic. Explicit schemes are generally hampered by a stringent restriction on the time step used in the computational procedure. For large regions simulation may be infeasible. Normally, implicit schemes do not have such restrictions and, therefore, can be applied using a significantly larger time step.

The method of comparison consists in applying both implicit and explicit codes to a seven and one half square mile area at Masonboro Inlet, North Carolina. Comparisons of surface elevations and depth averaged flows with prototype data are made for each scheme.

2. Theory--Equations of Fluid Flow. The hydrodynamic equations used in this work are derived from the standard three-dimensional Navier-Stokes equations. By assuming the vertical accelerations are small and the fluid is well mixed, and integrating the flow from the sea bottom to the water surface, the usual two-dimensional depth-averaged form of the equations of momentum and continuity are obtained:

#### MOMENTUM

$$\frac{\partial U}{\partial t} + \frac{U}{d} \frac{\partial U}{\partial x} + \frac{V}{d} \frac{\partial U}{\partial y} - fV + gd \frac{\partial \eta}{\partial x} = \frac{-gU}{C^2 d^2} (U^2 + V^2)^{1/2} + F_x \quad (1)$$

$$\frac{\partial V}{\partial t} + \frac{U}{d} \frac{\partial V}{\partial x} + \frac{V}{d} \frac{\partial V}{\partial y} + fU + gd \frac{\partial \eta}{\partial y} = \frac{-gV}{C^2 d^2} (U^2 + V^2)^{1/2} + F_y \quad (2)$$

#### CONTINUITY

$$\frac{\partial \eta}{\partial t} + \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = \xi \quad (3)$$

U, V: FLOW/UNIT WIDTH  
 $\eta$ : SURFACE ELEVATION

In these equations, U and V are the vertically integrated flows per unit of width at time t in the x and y directions, respectively;  $\eta$  is the water surface elevation with respect to the given datum; d is the water depth at (x,y,t); other terms are defined in the table of notations. As evident in the equations of motion, advection, coriolis force, bottom friction, rainfall, and wind forces are included. Two forms of finite difference solutions to equations (1), (2), and (3) are considered: an explicit and implicit formulation. The computational grid used in both

formulations is identical. A rectangular mesh is placed over the study area, and within each grid cell the following assumptions are made:

(1) the value of  $\eta$  is considered to be an average over a grid cell centered at  $x = M\Delta x$  and  $y = N\Delta y$  and at time  $k\Delta t$ ; (2) the value of  $U_{N,M}$  is given at the center of the lower cell face; and (3) the value of  $V_{N,M}$  is given at the center of the right-hand cell face (Figure 1).

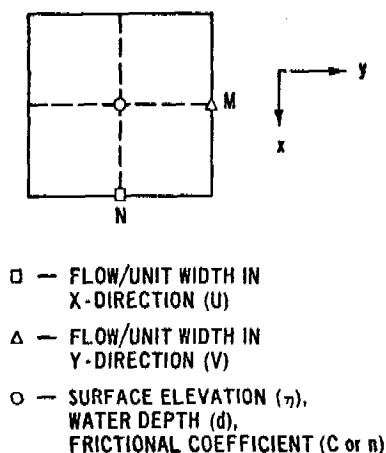


Figure 1. Cell definition

In addition, the water depth  $d$  and Chezy frictional coefficient  $C$  are also defined at the center of grid cells.

Explicit Solution Scheme. The explicit solution method used by Reid is a time-centered difference scheme involving a procedure of the "leap frog" type for computation of flow and water levels. The following notation will be used:  $k = k\Delta t$ ; angle brackets,  $\langle \rangle_1$  or  $\langle \rangle_2$ , to indicate that terms maintained in differential form are evaluated with centered difference expressions over one or two grid cells, respectively. Quantities which are not specified at a given spatial location are replaced by averaged values indicated by a bar, such as  $\bar{V}_{N,M+\frac{1}{2}}$ . Using these notations, applying centered differences in time and space, and

solving for  $U^{k+1/2}, V^{k+1/2}, \eta^{k+1}$  in terms of quantities at previous time levels, explicit expressions can be written as follows:

### MOMENTUM

$$U^{k+1/2} = \left[ U^{k-1/2} - \Delta t \frac{\bar{V}^{k-1/2}}{d^*} \left\langle \frac{\partial U}{\partial y} \right\rangle_2^{k-1/2} + \Delta t f \bar{V}^{k-1/2} - \Delta t g d^* \left\langle \frac{\partial \eta}{\partial x} \right\rangle_1^k + \Delta t F_x^k \right] / C_{ix} \text{ AT } (N, M + 1/2) \quad (4)$$

$$V^{k+1/2} = \left[ V^{k-1/2} - \Delta t \frac{\bar{U}^{k-1/2}}{d^*} \left\langle \frac{\partial V}{\partial x} \right\rangle_2^{k-1/2} + \Delta t f \bar{U}^{k-1/2} - \Delta t g d^* \left\langle \frac{\partial \eta}{\partial y} \right\rangle_1^k + \Delta t F_y^k \right] / C_{iy} \text{ AT } (N + 1/2, M) \quad (5)$$

### WHERE

$$C_{ix} = 1 + \frac{\Delta t}{d^*} \left\langle \frac{\partial U}{\partial x} \right\rangle_2^{k-1/2} + \frac{g n_x^2 \Delta t}{2.21(d^*)^{4/3}} W_x \quad (6)$$

$$W_x = \left[ (U^{k-1/2})^2 + (\bar{V}^{k-1/2})^2 \right]^{1/2} / d^* \quad (7)$$

$$C_{iy} = 1 + \frac{\Delta t}{d^*} \left\langle \frac{\partial V}{\partial y} \right\rangle_2^{k-1/2} + \frac{g n_y^2 \Delta t}{2.21(d^*)^{4/3}} W_y \quad (8)$$

$$W_y = \left[ (\bar{U}^{k-1/2})^2 + (V^{k-1/2})^2 \right]^{1/2} / d^* \quad (9)$$

$$d^* = \bar{\eta}^k - \bar{h} \quad (10)$$

### CONTINUITY

$$\eta^{k+1} = \eta^k - \Delta t \left[ \left\langle \frac{\partial U}{\partial x} \right\rangle_1^{k+1/2} + \left\langle \frac{\partial V}{\partial y} \right\rangle_1^{k+1/2} - \xi^k \right] \quad (11)$$

AT (N, M)

The calculations for each time step are divided into two halves; the flows are computed during the first half of the time step, and the results are used in the continuity equation to calculate the surface elevations during the second half.

Implicit Solution Scheme. To solve the governing equations implicitly, the same space-staggered scheme is used. The implicit code employs an alternating-direction technique whose calculations are divided into two parts. The first step, or  $\frac{1}{2}$ -cycle, consists in solving for  $\eta$  and  $U$  implicitly; the second  $\frac{1}{2}$ -cycle computes  $\eta$  and  $V$  implicitly. The omitted transport in each  $\frac{1}{2}$ -cycle is assumed constant for that step. Employing a centered difference operator to the momentum equation (1) and the continuity equation (3) along a grid line parallel to the  $x$ -axis, results in a system of linear algebraic equations whose coefficient matrix is tridiagonal. The form of the equations for the first  $\frac{1}{2}$ -cycle is given by:

#### MOMENTUM

$$\begin{aligned}
 U^{k+1/2} = U^{k-1/2} + \Delta t \left[ f \bar{V}^k - \frac{U^{k+1/2}}{d^*} \left\langle \frac{\partial U}{\partial x} \right\rangle_2^{k-1/2} \right. \\
 + F_x^k - \frac{\bar{V}^k}{d^*} \left\langle \frac{\partial U}{\partial y} \right\rangle_2^{k-1/2} - g d^* \left\{ \frac{\left\langle \frac{\partial \eta}{\partial x} \right\rangle_1^{k+1/2} + \left\langle \frac{\partial \eta}{\partial x} \right\rangle_1^{k-1/2}}{2} \right. \\
 \left. \left. + \frac{U^{k+1/2} + U^{k-1/2}}{2(\bar{C}^k)^2 (d^*)^3} \left( (U^{k-1/2})^2 + (\bar{V}^k)^2 \right)^{1/2} \right\} \right]
 \end{aligned} \quad (12)$$

AT  $(N, M + 1/2)$

#### CONTINUITY

$$\eta^{k+1/2} = \eta^k - \frac{\Delta t}{2} \left[ \left\langle \frac{\partial U}{\partial x} \right\rangle_1^{k+1/2} + \left\langle \frac{\partial V}{\partial y} \right\rangle_1^k \right] + \frac{\Delta t}{2} \epsilon^k \quad (13)$$

AT  $(N, M)$

$$\text{WHERE } d^* = \bar{\eta}^k - \bar{h} \quad (14)$$

Notations used in equations (7) and (8) are the same as those given in the description of the explicit formulation. The equations for the second  $\frac{1}{2}$ -cycle are similar to the above and are not presented.

Boundary Conditions. Various types of boundary conditions are permissible in the present system of computation for both explicit and

implicit codes. In both codes, boundary conditions at the tide computation boundary (open boundary conditions) are accomplished by setting the water levels,  $\eta_{N,M}$ , as prescribed by input tables. Flow rates may be specified instead of water levels. All additional boundary conditions relate the normal component of flow at the boundary to the state of the water level at the boundary.

Water-Land Boundaries. Such boundaries are prescribed along cell faces, hence this condition is handled by specifying  $U = 0$  or  $V = 0$  for those cells where impermeable boundaries exist. In estuarine systems with large areas of low-lying terrain and a significant tidal range, many areas alternately dry and flood with each tidal cycle. This behavior is simulated by making the location of the land-water boundary a function of the current value of the total water depth. By checking the water level in adjacent cells relative to the ground elevation, a determination is made as to the possibility of inundation. If flooding is possible, the boundary face is treated as open and computations for  $\eta$ ,  $U$ , and  $V$  are made for that cell. The drying of cells is simply the inverse process.

Subgrid Barriers. Subgrid barriers are defined along cell faces and are of three types: exposed, submerged, and overtopping. One characteristic of such barriers is that the surface elevation is computed at the center of cells on either side of the barrier. The treatment of these barriers in the explicit code can be found in Masch (1973). The following discussion is limited to the way in which these conditions are simulated in the implicit code. Exposed barriers are handled by simply specifying a no-flow condition across the cell face. This type of barrier is used to describe dykes, jetties, and similar features which are impermeable and usually of width much less than  $\frac{1}{2}$  the spatial grid step. Submerged barriers are used to simulate flows across such barriers as submerged reefs, spoil banks, pipe lines, etc. The water level on each side of a submerged barrier must always exceed the barrier crest elevation. The flow over a submerged barrier can be controlled in a manner similar to that used by Masch, but experience

has shown that by defining a special Chezy coefficient for the barrier face, the flow over the barrier can be simulated without introducing unwanted transients due to the use of the submerged weir equation.

Overtopping barrier is a terminology used to distinguish a barrier which can be submerged during one portion of the tidal cycle and totally exposed in another. Masch used a broad-crested weir formula to describe the overtopping nature of the flow and then the submerged weir formula when appropriate. Since a larger time step is used in the implicit code and the duration time of overtopping is short, a Chezy formulation is again used to simulate the flow across the barrier. When the barrier is exposed a very small Chezy coefficient (high friction) is used to "stop" the flow. When overtopping occurs, the coefficient is increased to a specified maximum as a function of the water-level over the barrier. As the water level decreases at a later time, the coefficient is decreased accordingly.

Numerical Stability. For an explicit solution scheme, the grid size and computational time step are related through a stability criterion. The criterion associated with the explicit scheme presented here is given by the relation

$$\Delta t \leq \frac{\Delta s}{\sqrt{2gd_{\text{MAX}}}} \quad (15)$$

where  $\Delta s$  is the mesh size and  $d_{\text{MAX}}$  is the maximum water depth used in the model. This approximate condition was derived from expressions obtained by linearizing the problem. When the non-linear terms are included, it can be expected that the time step will require further reduction.

Considering the linearized implicit equations, it can be shown that the difference scheme is unconditionally stable. In other words the space and time steps may be chosen to meet required accuracy in representing topographic features and external forcing functions. The inclusion of the non-linear advective terms (of the form  $U \frac{\partial U}{\partial x}$ ,  $V \frac{\partial U}{\partial y}$ )



into the implicit scheme results in inherent instabilities. All derivatives in the basic equations are approximated with centered differences over a single grid space with the exception of the advective terms, which are computed over two grid cells. Oscillations of the water level at a grid point (period  $4\Delta t$ ) may occur and grow unbounded.

A scheme which proved capable of eliminating these instabilities was the use of a recursive digital filter of the form

$$\eta^{k+1} = a\eta^{(c)k+1} + b\eta^k + c\eta^{k-1} \quad (16)$$

where  $\eta^{(c)}$  represents the computed water level and  $\eta$ , the water level value used in further computations; coefficients  $a, b, c$  are chosen to filter out oscillations of period  $4\Delta t$  (corresponds to  $\frac{1}{2}$  the Nyquist frequency) and smaller, while permitting the longer period wave motion to remain undisturbed. The coefficients of the digital filter must also be chosen in such a way as to maintain stability of the filter. By applying the linearized system with and without a filter it was demonstrated that filtering does not affect the results. For applications presented in this paper, values of  $a = 0.6$ ,  $b = 0.3$ , and  $c = 0.1$  were selected.

An additional instability, which is termed a "secondary flow" phenomena, may also occur. A discussion of this problem was presented by Vreugdenhil (1973). The scheme normally employed to eliminate this instability is the inclusion of terms in the momentum equations of the form

$$\epsilon \left[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right] \quad (17)$$

Such a form is referred to as an eddy-viscosity term and is generally taken as a representation of the effective-stress in vertical planes. These terms are usually neglected but are very important when the flow has a strong tendency to converge or diverge at various locations within the system.

3. Application to Masonboro Inlet, North Carolina--Computational Grid. A map of the Masonboro Inlet system is shown in Fig. 2. A grid size of 300 feet was adapted resulting in a mesh of dimensions 41 X 57. The computational water points are 1721 in number. Tidal elevations obtained from prototype data taken in a survey on 12 September 1969 are imposed at boundary lines denoted by circled numbers. The bathymetry at Masonboro Inlet for this period of time was also surveyed; available boat sheets were digitized for the computational area. The frictional coefficients are defined by assigning number codes to the various types of terrain and applying known values of Manning's n and the relationship:

$$C = \frac{1.49}{n} d^{1/6} \quad (18)$$

Since the prototype data was taken on a calm day, no wind stress was applied in the numerical model.

The jetty system protruding from the outer barrier island is composed of a weir section, elevation 2' above datum and 1000' in length, extending from the outer island to near the bend in the jetty. The remainder of the structure is impermeable. Exposed barriers were used to represent very narrow strips of high land in the marsh area behind the barrier islands. The outer model boundaries are set at a distance of 10,000 feet from the inlet throat to minimize effects from inherent problems in handling all the terms in the equations at the input boundaries.

Comparison of Results. Although excellent results were obtained by both codes at gages located throughout the system, for brevity, only results at the three locations depicted in Fig. 2 will be presented for comparison. Figure 3 shows the degree to which the numerical codes simulate the prototype tides. The computations were begun at 1330 EST and a fold-over occurred at 2000 EST, equating data at this hour with that taken at 0730 EST. This procedure may cause some discrepancy in the results but is required since the models must spin up from rest at low tide. Both models describe the tides equally well. The discrepancy at gage 2 in the ebb phase may have resulted from two

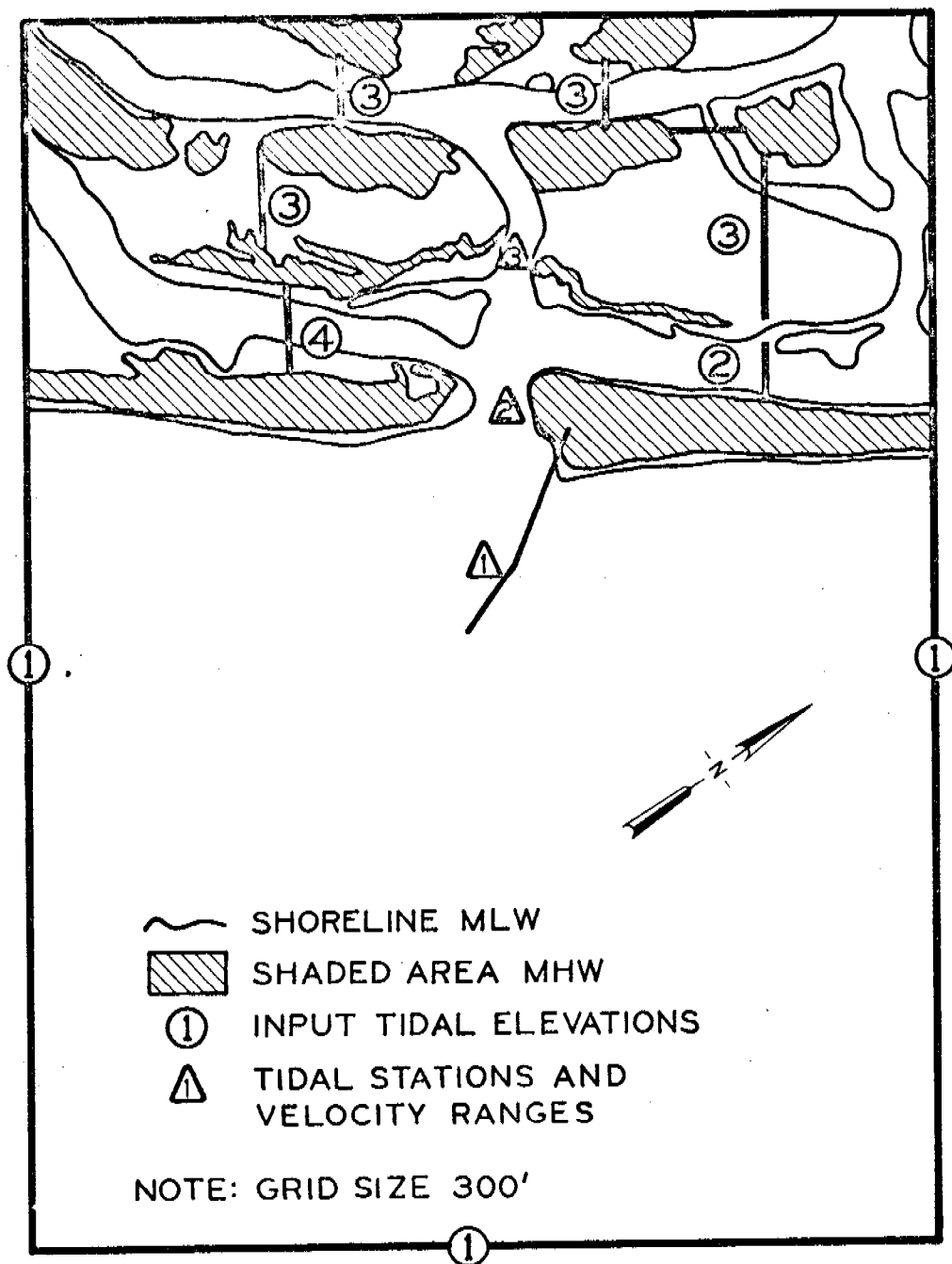


Figure 2. Aerial extent of computational grid for Masonboro Inlet, North Carolina

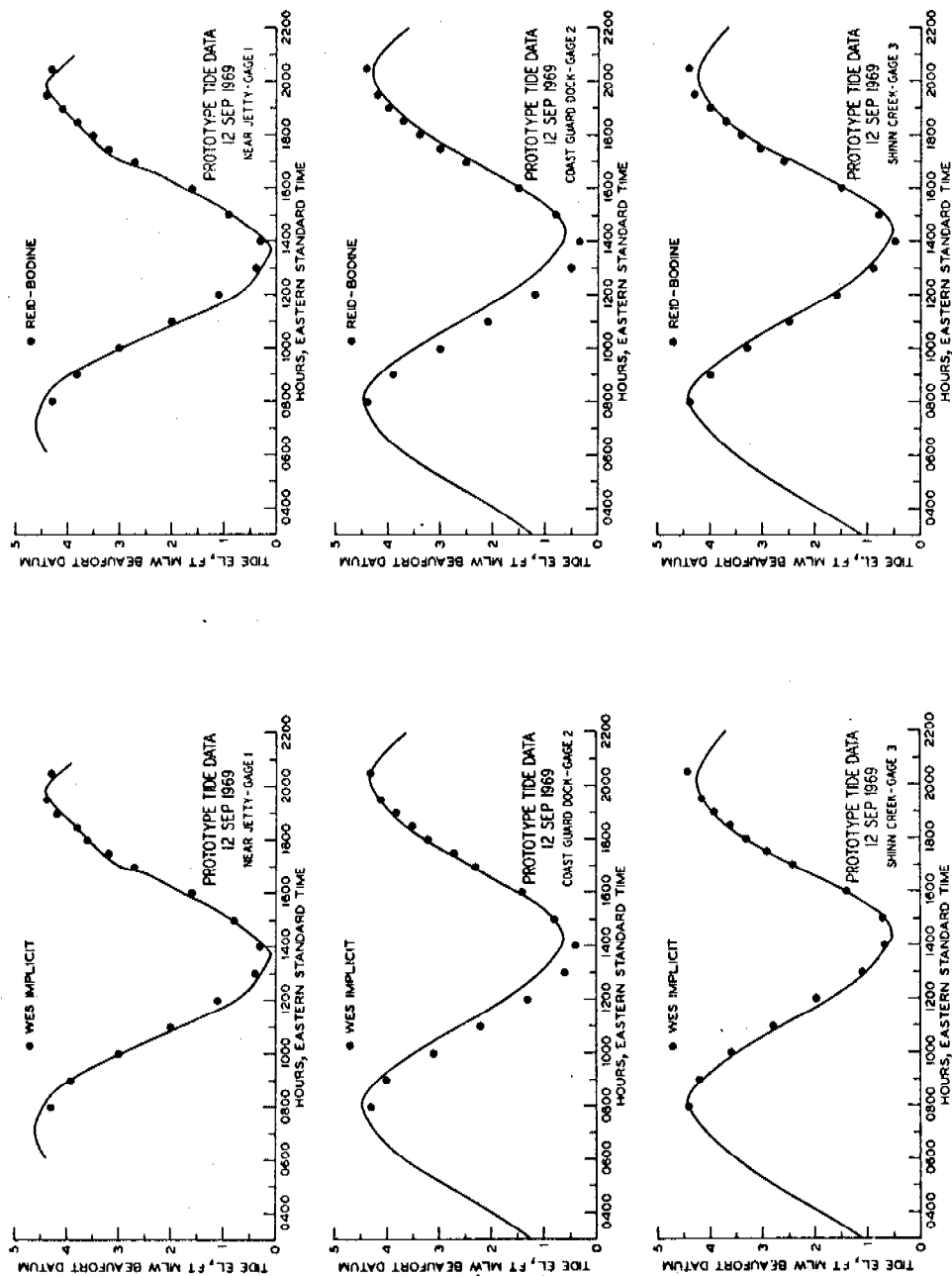


Figure 3. Comparison of surface elevation agreement with prototype

problems: the prototype tide gage was located within a marina behind the northern barrier island and there was some question as to the phasing of the tidal input specified at boundary 2.

Figure 4 shows the tidal velocities calculated in the numerical models relative to prototype velocity measurements taken at three depths: surface, mid-depth and bottom. Recalling that the models produce a depth averaged flow rate, velocities are obtained by dividing the flow by the local water depth. Again, good agreement is obtained in both models.

Figures 5-6 display sample circulation patterns at flood stage (5 Hrs = 1830 EST) and ebb stage (10 Hrs = 1100 EST). The arrowheads indicate direction of flow and their length is proportional to the magnitude of the flow rate.

4. Comparison Statistics and Conclusions. Table 1 below relates the computer run time required for the Masonboro simulation for both numerical models.

TABLE 1. COMPARISON STATISTICS

<u>GRID</u>		
NUMBER OF WATER POINTS		1721
SPATIAL STEPSIZE (FT)		300
<u>SOLUTION</u>		
	$\Delta t$	RUN*
	(SEC)	(MIN)
EXPLICIT	3	60
IMPLICIT	90	4

\* BASED ON 18 PROTOTYPE HOUR  
SIMULATION ON A CDC 7600.

The stability criterion associated with the explicit scheme predicts a 5.2 second time step is necessary for the range of water depths appearing in the system. However, time steps of 5 sec and 4 sec were tried and

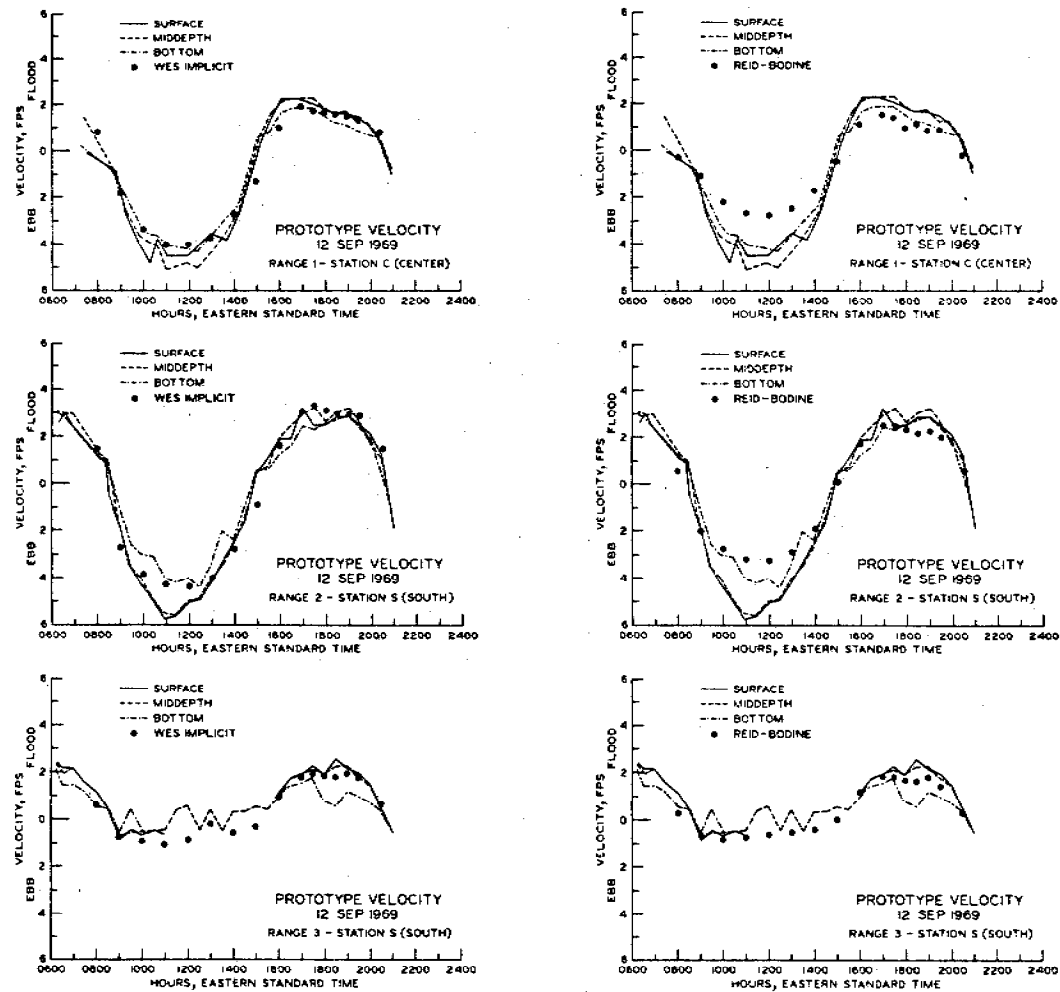


Figure 4. Comparison of velocity agreement with prototype

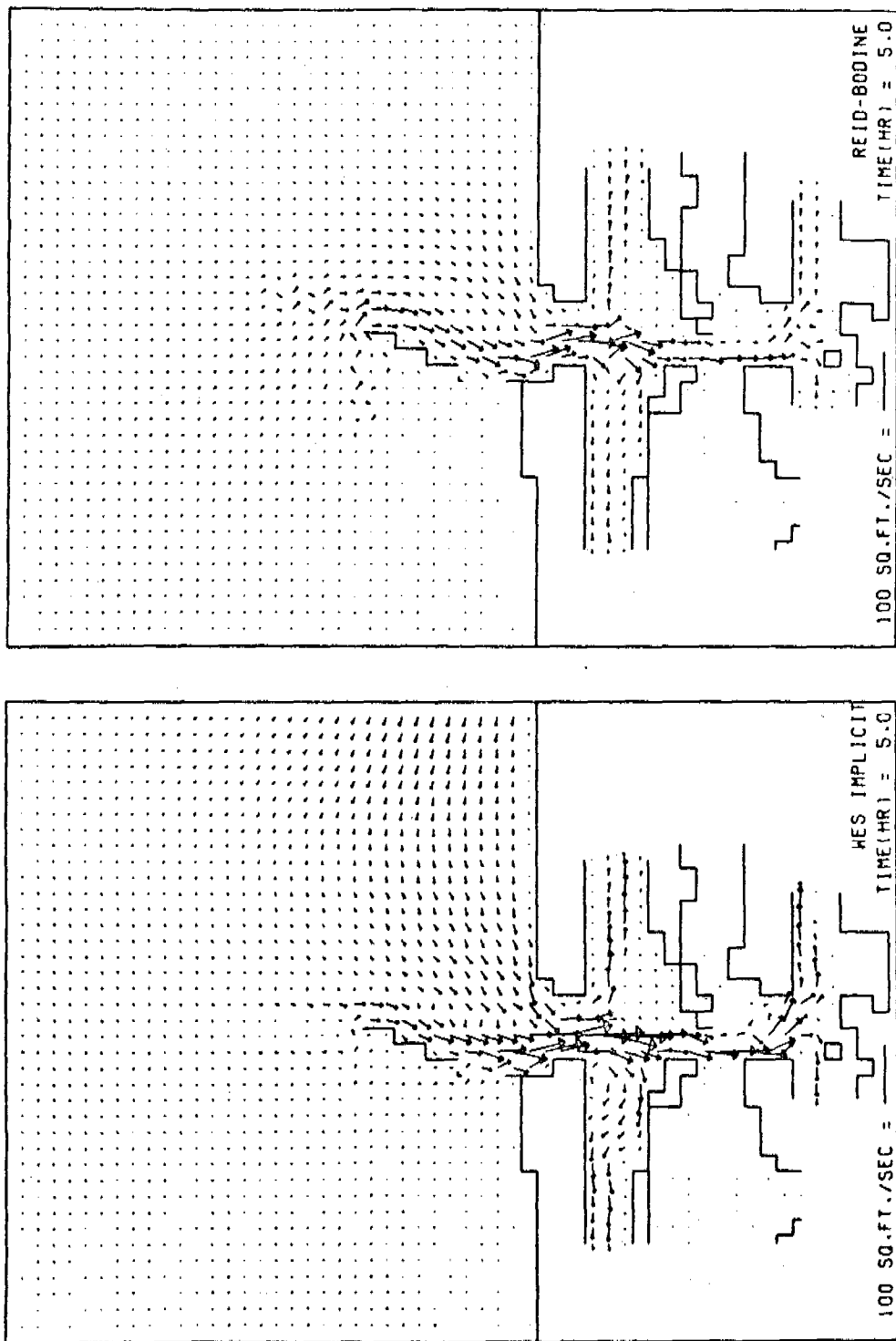


Figure 5. Circulation patterns at flood stage

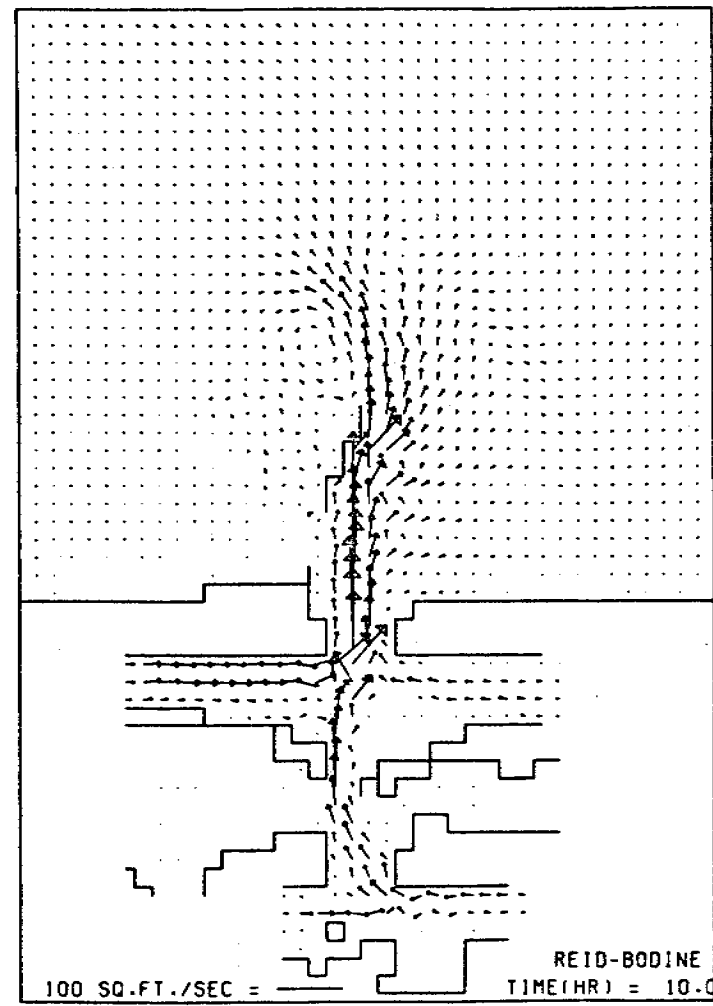
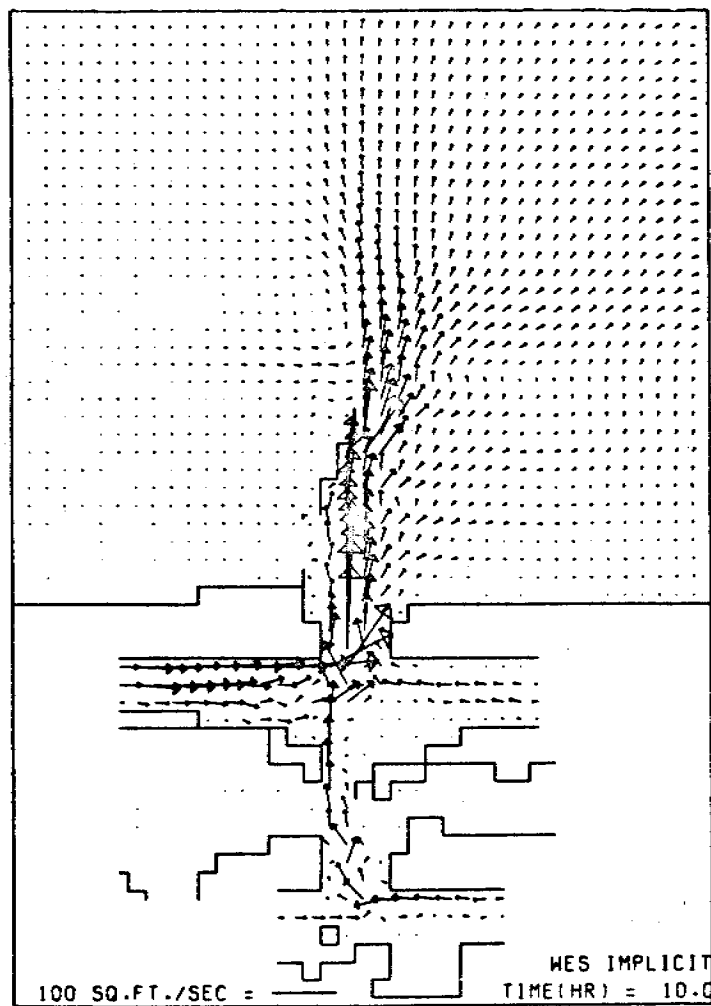


Figure 6. Circulation patterns at ebb stage



instabilities resulted. A stepsize of 3 sec was found to be stable for the entire simulation. The implicit scheme was run for time steps of 45, 90, and 180 seconds. Results for the two smaller time steps were practically identical. Results for  $\Delta t = 180$  compared favorably with the prototype but some discrepancies in phasing were noted. Results for  $\Delta t = 90$  seconds have been presented. Note that  $\Delta t$  for the implicit scheme is the time for a complete cycle, that is, U and V are computed once and  $\eta$ , twice. In the explicit scheme, U, V and  $\eta$  are computed once in a time step of 3 seconds. The speed of the implicit scheme can be expressed by the relationship

$$I \approx .5 \frac{\Delta t_i}{\Delta t_e} \quad (19)$$

where  $\Delta t_i$  and  $\Delta t_e$  are the time steps of the implicit and explicit schemes, respectively, and I is the execution speed of the implicit model relative to the explicit model. In this application a 15:1 ratio in execution time was noted.

The results of this study demonstrate that the implicit scheme presented herein can reliably simulate the tidal hydrodynamics of a complicated inlet system. The WES implicit model is considerably more economical to apply and should prove to be most beneficial in applications to estuarine systems as well as other problems which require simulation of long period wave motion.

## REFERENCES

1. Butler, H. Lee and Durham, D. L., (1975), "Applications of Numerical Modeling to Coastal Engineering Problems," presented at the 1975 Army Numerical Analysis and Computers Conference, St. Louis, Mo., Feb 1975 (to be published in 1976 proceedings).
2. Leendertse, J. J., (1970), "A Water-Quality Simulation Model for Well-Mixed Estuaries and Coastal Seas: Volume I, Principals of Computation," The Rand Corporation, RM-6230-RC, Feb 1970.
3. Leendertse, J. J., (1971), "A Water-Quality Simulation Model for Well-Mixed Estuaries and Coastal Seas: Volume II, Computational Procedures," The Rand Corporation, R-708-NYC, April 1971.
4. Masch, F. D., Brandes, R. J. and Reagan, J. D., (1973), "Simulation of Hydrodynamics in a Tidal Inlet," Water Resources Engineers, Inc., Technical Report to the Army Coastal Engineering Research Center, Feb 1973.
5. Reid, R. O. and Bodine, B. R., (1968), "Numerical Model for Storm Surges in Galveston Bay," Journal of Waterways and Harbors Division, Proceedings, ASCE, Vol. 94, No. WW1, Feb 1968.
6. Vreugdenhil, C. B. (1973), "Secondary-Flow Computations," Delft Hydraulics Laboratory, Publication No. 114, Nov 1973.

### NOTATIONS

$C$	Chezy frictional coefficient
$d$	Total water depth
$d_{MAX}$	Maximum water depth at any location and time in the system
$f$	Coriolis parameter
$F_x, F_y$	External forcing functions
$g$	Acceleration of gravity
$h$	Land surface elevation
$k$	Time increment counter
$M, N$	Indices denoting spatial increments in the x and y direction
$n$	Manning Frictional coefficient
$n_x, n_y$	Averaged Manning's coefficient
$t$	Time
$U, V$	Integrated horizontal velocity components
$x, y$	Cartesian coordinates
$\epsilon$	Eddy-viscosity coefficient
$\eta$	Water surface elevation with respect to given datum
$\xi$	Rainfall minus evaporation
$\Delta x, \Delta y$	Spatial increment
$\Delta s$	General spatial increment
$\Delta t$	Time increment



# A MODEL FOR LANDSLIDE GENERATED WATER WAVES

Donald C. Raney  
AME Department  
University of Alabama  
University, Alabama 35486

H. Lee Butler  
U. S. Army Waterways Experiment Station  
Hydraulics Laboratory, Wave Dynamics Division  
Vicksburg, Mississippi 39180

ABSTRACT. A numerical model is developed for simulating the development and propagation of landslide generated water waves in reservoirs. The numerical model is based upon a finite difference representation of the depth averaged hydrodynamic equations. The landslide is formulated as a moving boundary condition, propagating into the reservoir and accelerating the fluid due to physical displacement and viscous drag. Arbitrary reservoir geometry and landslide parameters can be considered. The numerical model results are compared with experimental results obtained on a 1:120 undistorted scale physical model of Libby Dam and Lake Koocanusa in Montana. Landslides were considered reflecting a wide range of landslide volumes and velocities. The wave heights predicted by the numerical model are in good agreement with the wave heights observed in the physical model.

1. INTRODUCTION. There are many serious problems associated with rockfalls or landslides into bays, lakes, reservoirs, fjords and rivers. These problems are becoming increasingly important due to expanded use of these bodies for recreational purposes and the increased industrial and residential development along the shores. Some areas of concern created by potential slide areas are: loss of life, damage to shoreside structures and boats, overtopping of dams by surge or waves with resulting damage to the dam and spillways, failure of dams with resulting large scale flooding, upstream flooding due to river blockage and loss of usage of the water body due to restrictions imposed by the final position of the slide material. Examples of the occurrence of each of these can be found in the literature and vividly indicate the extent of the potential problem.

2. BACKGROUND REVIEW. Some attention has previously been directed toward obtaining a qualitative and quantitative understanding of the probability of occurrence and the characteristics of water waves generated by rockfalls or landslides into reservoirs. Most laboratory and theoretical

investigations, however, have been two dimensional studies in an attempt to determine the fundamental relationships between the geometry and speed of a rockfall or landslide and the characteristics of the water waves generated by this mechanism (5, 7, 8).

In 1971 the Hydraulics Laboratory at the U. S. Army Corps of Engineers Waterways Experiment Station (WES) constructed a 1:120 undistorted model of Libby Dam and Lake Koocanusa and conducted tests to determine generated wave heights and characteristics resulting from the sliding of individual rock ribs into the reservoir (1). Libby Dam, a 420 ft. (128 m) high concrete gravity structure on the Kootenai River in western Montana, is flanked by high rock slopes extending several thousand feet upstream from the structure. The rock is predominately bedded and jointed. Several prominent rock ribs form possible rock slide zones.

The present study was directed toward developing numerical methods for predicting the effects of landslide generated water waves in reservoirs. The previous physical model tests were used to provide comparative data.

3. THE NUMERICAL MODEL. A two dimensional approach which possesses a pseudo three dimensional effect was utilized in the numerical investigation. The vertical component of velocity is neglected and the governing hydrodynamic equations are integrated over the water depth. An average two dimensional flow field is obtained but three dimensional geometry can be considered. This basic approach has been used by several authors such as Hansen (2), Leendertse (4) and Platzman (6).

The rectangular coordinate system used is located in the plane of the undisturbed water surface as shown in Figure 1. The equations of motion and the equation of continuity are written as follows:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + g \frac{\partial \eta}{\partial x} = R_x + L_x \quad (1)$$

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + g \frac{\partial \eta}{\partial y} = R_y + L_y \quad (2)$$

and

$$\frac{\partial \eta}{\partial t} - \frac{\partial \zeta}{\partial t} + \frac{\partial}{\partial x} [(h + \eta) u] + \frac{\partial}{\partial y} [(h + \eta) v] = 0. \quad (3)$$

In these equations  $u$  and  $v$  are velocity components,  $\eta$  is the water level displacement relative to the initial reservoir

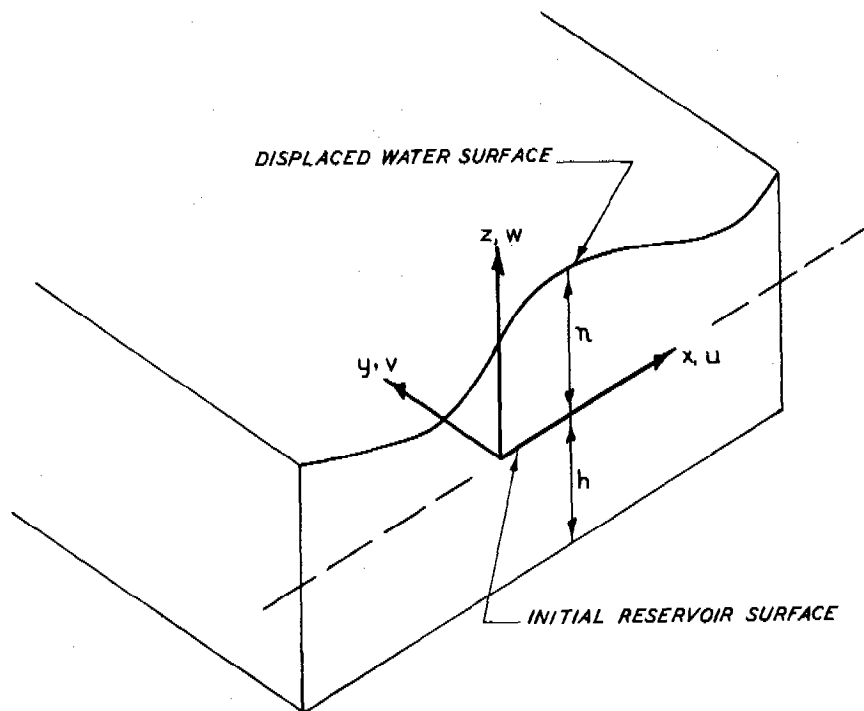


Figure 1. Coordinate system for problem formulation

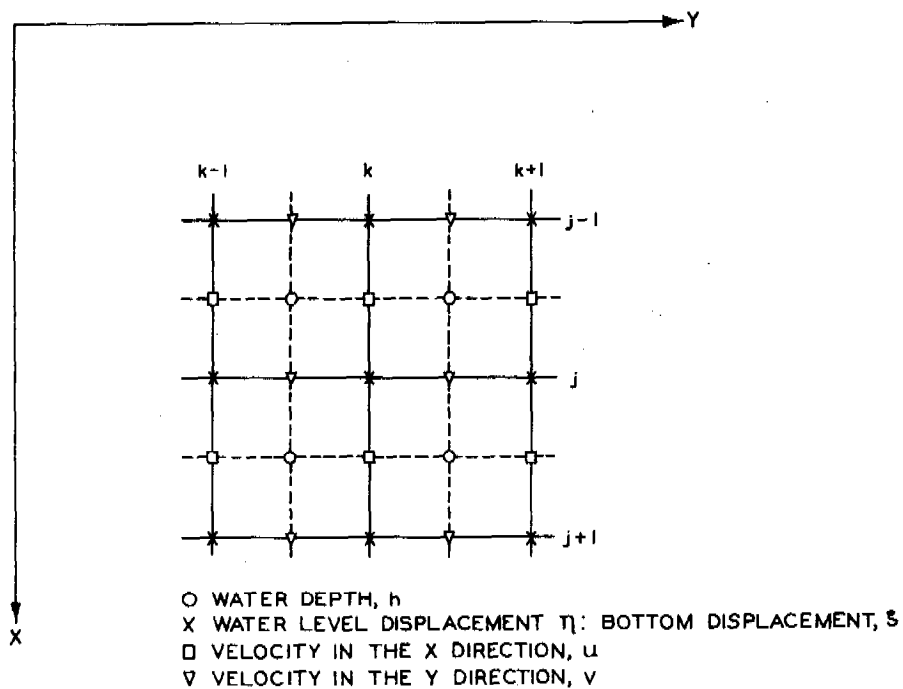


Figure 2. Grid system and variable definition location

surface,  $h$  is the undisturbed reservoir depth,  $\zeta$  is the vertical bottom deformation created by the landslide material,  $R$  is the bottom friction and  $L$  is a direct acceleration effect of the landslide on the fluid. Additional terms, Coriolis effect, horizontal diffusion, and wind stress could be considered in the equations of motion; however, these terms were neglected in this investigation. The continuity equation has been obtained by integrating across the water depth and applying kinematic and dynamic boundary conditions at the surface and bottom of the reservoir. The bottom friction terms are represented using a modification of the normal formulation of friction in terms of the Chezy coefficient. This formulation is necessary to properly account for the bottom friction where the moving landslide represents the bottom boundary.

$$R_x = \frac{g (V_x - u) [(V_x - u)^2 + (V_y - v)^2]^{\frac{1}{2}}}{C^2 (h + \eta)} \quad (4)$$

$$R_y = \frac{g (V_y - v) [(V_x - u)^2 + (V_y - v)^2]^{\frac{1}{2}}}{C^2 (h + \eta)} \quad (5)$$

The velocity components  $V_x$  and  $V_y$  are the components of the landslide velocity.

To solve the governing equations a finite difference approximation of the equations and an implicit-explicit alternating direction technique is employed. A space staggered scheme is used in which velocities, water level displacement, bottom displacement, and water depth are described at different locations within a grid cell as shown in Figure 2. A double-time-step operation is used in such a manner that the terms containing space derivatives are generally taken as alternating forward and backward. The first step in the calculation consists of computing  $u$  and  $\eta$  implicitly and  $v$  explicitly, advancing from time  $n \Delta t$  to  $(n + \frac{1}{2}) \Delta t$ . The second step computes  $\eta$  and  $v$  implicitly and  $u$  explicitly, advancing from time  $(n + \frac{1}{2}) \Delta t$  to  $(n + 1) \Delta t$ . Central differences are used for evaluating all derivatives in the governing equations. This method of solution has been discussed in detail by Leendertse (4).

Three types of boundaries are involved in the calculations. These are the solid boundaries at fixed coastlines, the fictitious open boundaries arising from the need to truncate the region of computation and the time dependent boundary between the landslide surface and the water in the reservoir.



A condition of complete reflection is adopted at solid boundaries. The condition can be written as

$$\bar{v} \cdot \bar{n} = 0 \quad (6)$$

at solid boundaries, where the  $\bar{n}$  denotes a unit vector normal to the boundary.

The applicable condition at fictitious open boundaries is more difficult to specify. The total transmission of the wave is the physical requirement at this boundary; however, this cannot be rigorously achieved without computation beyond the boundary. As an approximation to the desired physical requirement, the wave profile is simply assumed to travel without change of form across the last interior grid cell.

The landslide is represented by a time dependent vertical deformation of the bottom of the reservoir plus additional terms to represent the effect of the landslide due to viscous and inertia forces. The bottom deformation propagates into specified regions of the reservoir at the average speed of the landslide with the deformation at any particular location increasing from zero to a maximum value according to a specified time-displacement relationship. For those portions of the bottom of the reservoir through which the landslide passes but which do not experience a net change in ground elevation, the deformation is allowed to return to zero at a specified rate. The handling of the landslide condition is illustrated in Figure 3. The direction, extent, and magnitude of the bottom deformation is determined by knowledge of assumptions concerning the path and final disposition of the particular landslide. The water in the reservoir experiences an acceleration due to the force exerted by the landslide at the time dependent boundary between the water and landslide. This force per unit mass consists of a component due to the vertical displacement of the water by the slide, a component due to the bottom friction between the landslide and the water, and a pressure drag exerted on the water by the front of the moving landslide. The pressure drag at the leading edge of the slide can be represented by

$$(F_p)_x = C_D \left(\frac{1}{2}\rho\right) (V_x - u) [(V_x - u)^2 + (V_y - v)^2]^{\frac{1}{2}} A_z \quad (7)$$

where  $C_D$  is a pressure drag coefficient and  $A_z$  is an effective vertical cross-sectional area of the slide. The force per unit mass can then be considered as:

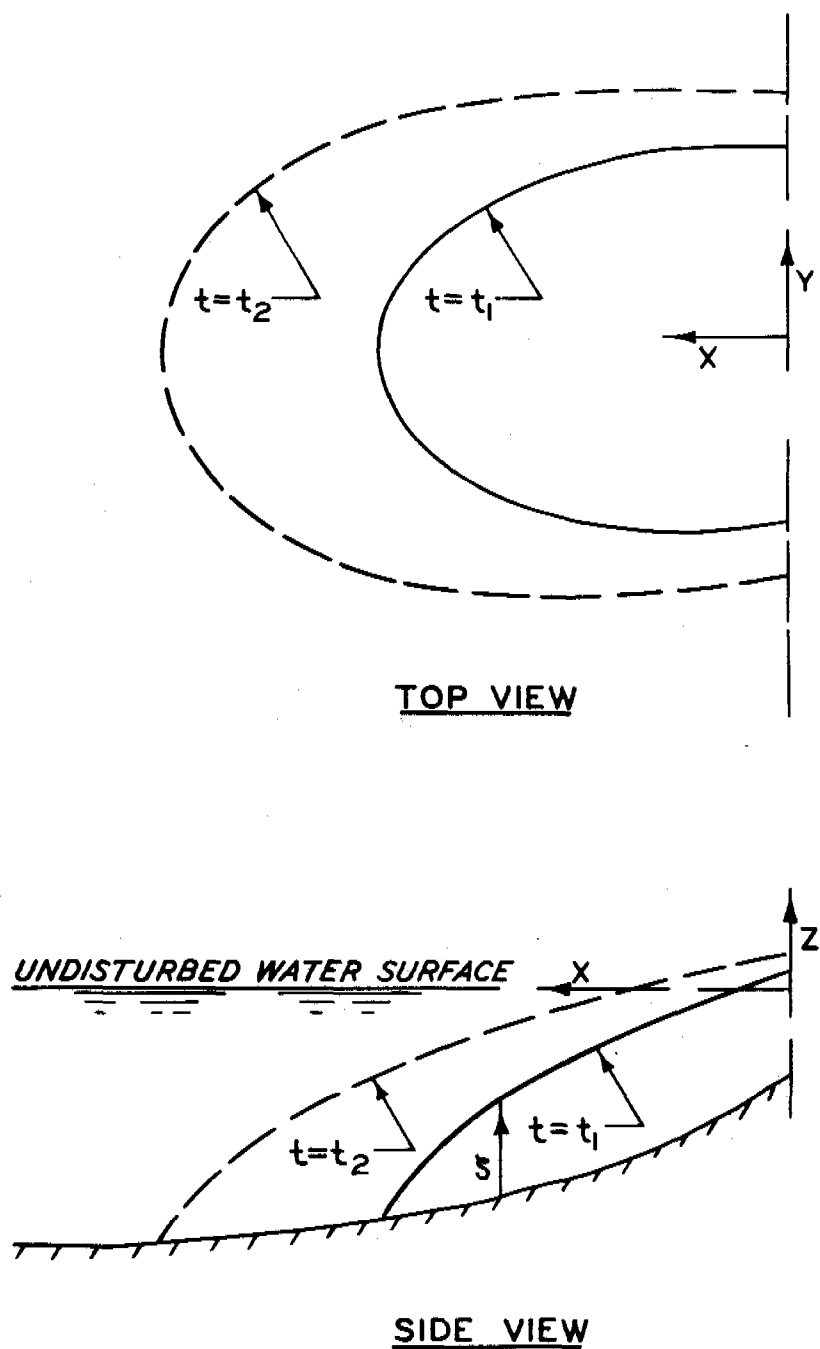


Figure 3. Position of the landslide at arbitrary times

$$\frac{(F_P)_x}{\text{MASS}} = \frac{\frac{1}{2} (C_D A_z) (v_x - u) [(v_x - u)^2 + (v_y - v)^2]^{\frac{1}{2}}}{A_c h}$$

where  $A_c$  is the grid cell area and  $h$  is the water depth. The pressure drag contribution per unit mass can then be considered as:

$$L_x = \beta (v_x - u) [(v_x - u)^2 + (v_y - v)^2]^{\frac{1}{2}} \quad (8)$$

$$L_y = \beta (v_y - v) [(v_x - u)^2 + (v_y - v)^2]^{\frac{1}{2}} \quad (9)$$

$$\text{where } \beta = \frac{C_D}{2h} \left( \frac{A_z}{A_c} \right) \quad (10)$$

A representation of the landslide is allowed to propagate into the numerical representation of the reservoir, accelerating the fluid due to physical displacement, viscous effects, and pressure drag effects. The resulting waves propagate across the reservoir in accordance with the governing equations. The wave height and velocity components are calculated for each grid cell at the end of each one half time step.

4. BRIEF DISCUSSION OF PHYSICAL MODEL TESTS. The numerical results were compared with experimental data from a physical model of Libby Dam and Lake Koocanusa (1). A site map in Figure 4 shows the topography of the steep rock slopes upstream of Libby Dam. Potential landslide zones upstream of the left abutment are denoted as rock ribs 909, 914, 923, 927 with reference to the stationing along Montana State Highway 37. The area covered by the hydraulic model study is shown by heavy dotted lines. The locations of two prehistoric landslides denoted as 925 slide and 930 slide are also shown in Figure 4. The Libby Dam hydraulic model was constructed to a linear scale of 1:120, model to prototype. An undistorted scale was used to insure accurate reproduction of wave heights, wave period and runup. The dimensions of the model were 57 ft (17.4 m) long, 40 ft (12.2 m) wide and about 5 ft (1.5 m) deep. The maximum elevation reproduced in the model was 2700 ft (823 m) msl with an adjustable mechanical inclined plane to support the landslide material above this elevation. A range of possible landslide velocities was considered at each potential slide location. These velocities varied between 37 fps (11.3 m/sec) and 192 ft/sec (58.5 m/sec).

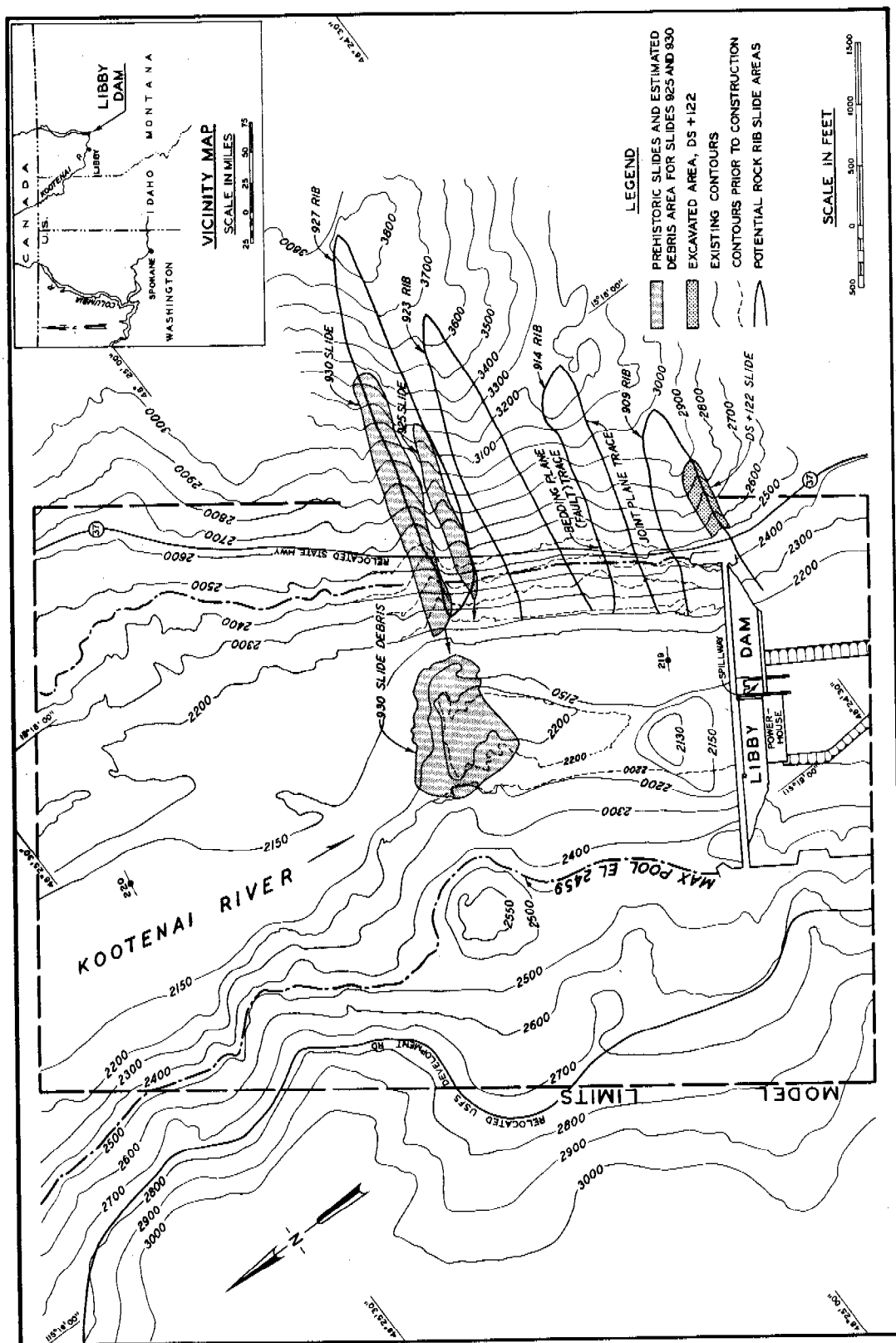


Figure 4. Site map

Wave heights at selected locations in the model were measured using direct contact electrical wave-height gages. The gage locations are shown in Figure 5. The runup distances on the sides of the model were observed and the water overtopping the dam was collected and measured.

Figure 6 shows the physical model ready for a test run. The landslide material used in the model tests was 1/18 cu. ft. (0.028 cu m) bags filled with lead and iron ore. Photographs showing the final position of the slide were taken and the approximate final contours of the reservoir were determined after each test. A typical set of these data are shown in Figures 7 and 8 for Run 87.

5. THE NUMERICAL CALCULATIONS. The numerical model required that a rectangular grid of mesh cells be established as well as the computational boundaries be established, and appropriate information be defined at discrete points in the reservoir. The grid size is selected to obtain the desired spatial resolution. Initial data include defining the reservoir depth and a Chezy coefficient at each grid line. The time step is chosen using as a general requirement:

$$\Delta t < \frac{\Delta x}{(gh)^{\frac{1}{2}}} \quad (11)$$

This relation restricts movement of the water wave to less than one grid space per time step.

Defining the landslide characteristics is the most critical aspect of the model. It is necessary to know or assume the volume of the slide material, the average velocity at which it moves, its path through the water, the general shape of its leading face, a time-vertical displacement relationship for the slide and the final disposition of the slide in the reservoir. In a general investigation the use of this model would require a parametric study. Available for this study were experimental data from a physical model study so that these parameters were known or could be approximated to a reasonable degree of accuracy. This study then reflects the degree to which the mathematical model can represent the reservoir conditions if the slide characteristics are reasonably well defined.

A time step of 1 sec and a spatial grid size of 80 ft (24.2 m) were used in the calculations. The pressure drag parameter  $\beta$  was varied between 0.005 and 0.0005 ft<sup>-1</sup>.

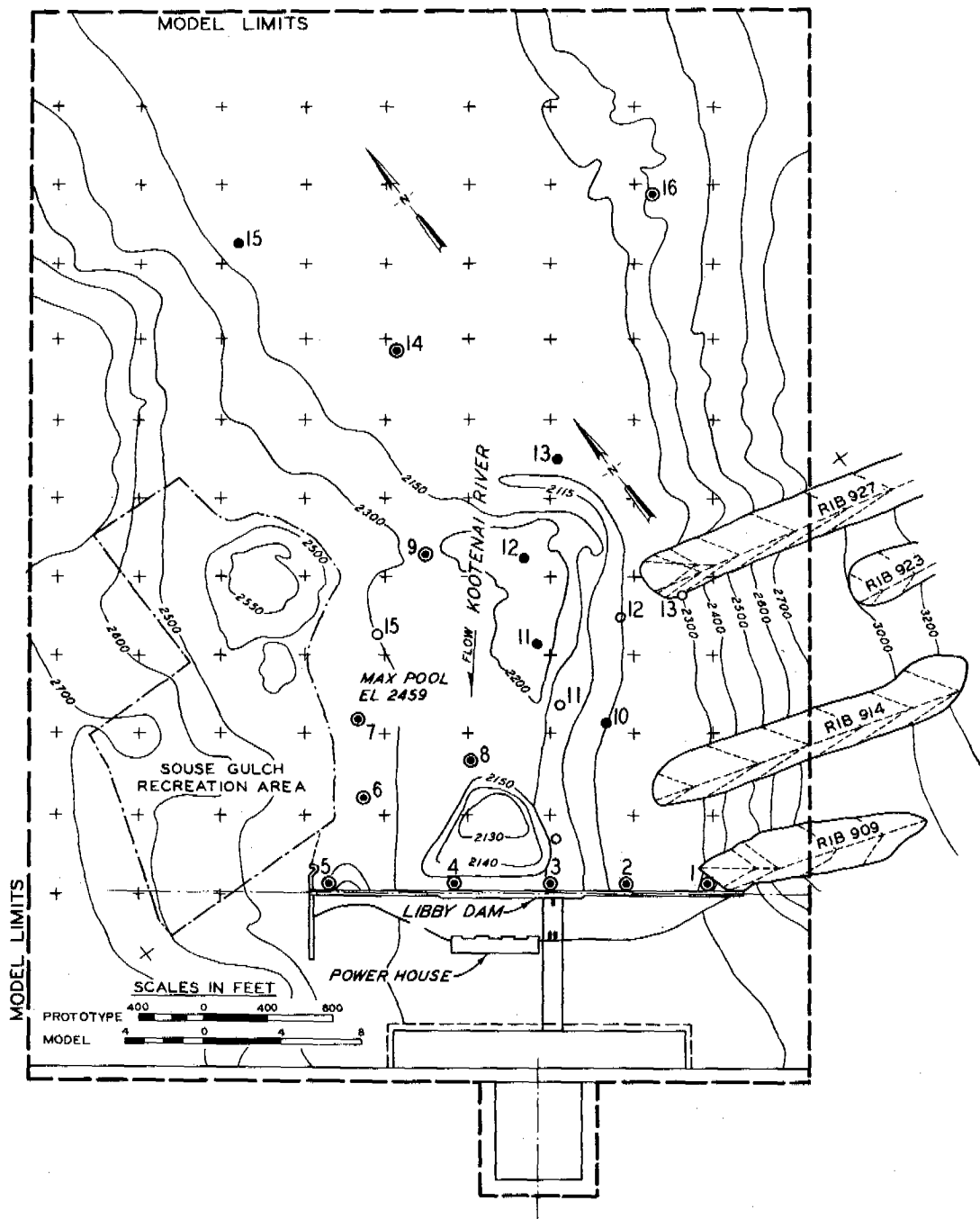


Figure 5. Location of wave gages

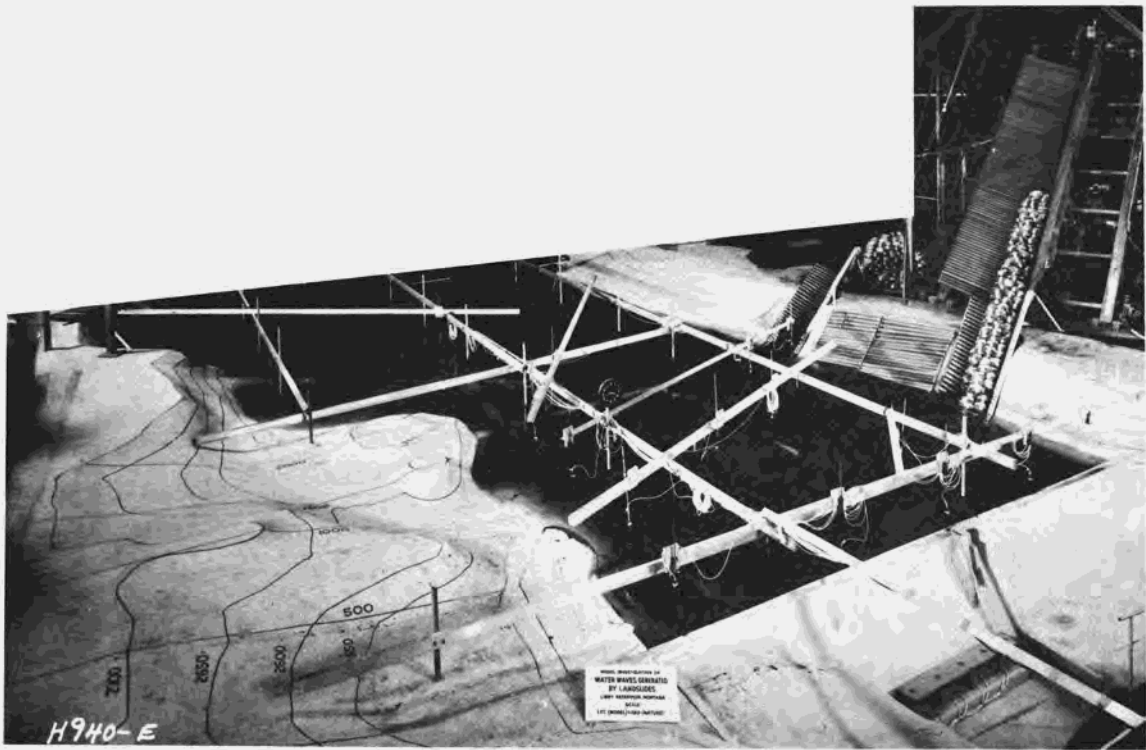


Figure 6. Physical model ready for test run

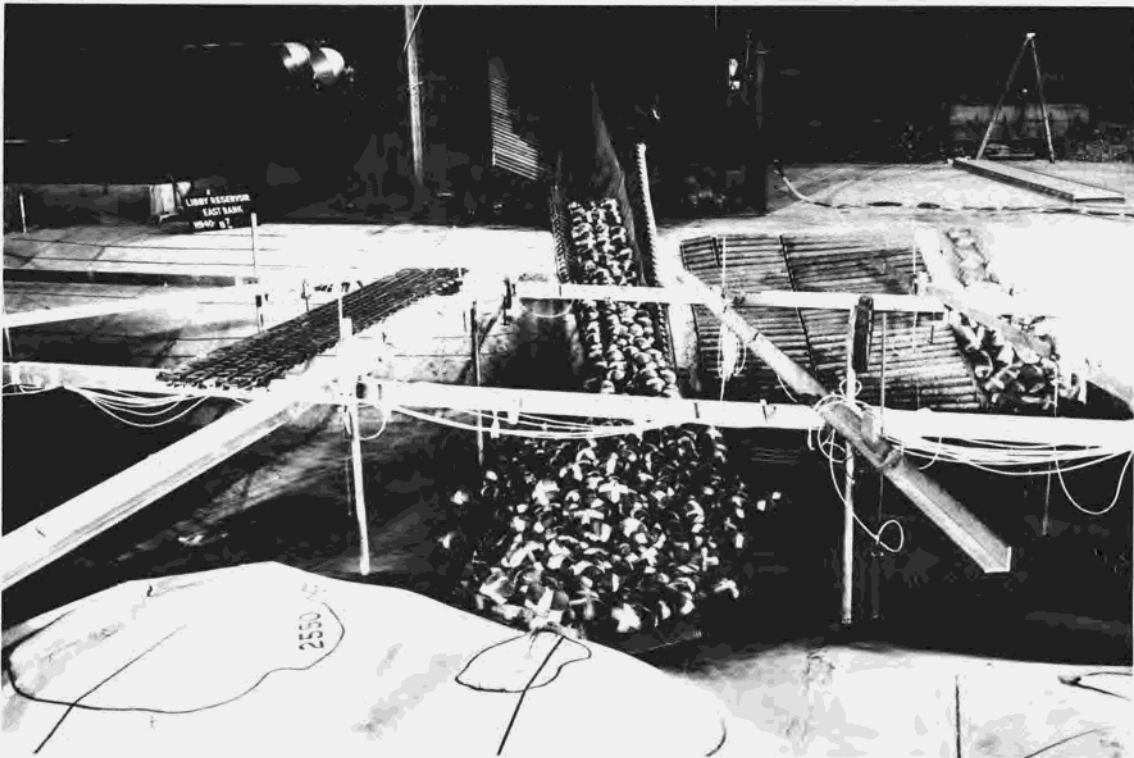


Figure 7. Final slide position for run 87

6. COMPARISON OF MATHEMATICAL MODEL AND PHYSICAL MODEL RESULTS. To provide some degree of verification of the mathematical model, results are compared with data obtained from physical model tests. Five landslide conditions from the physical modeling program were selected for comparison. Included in the five slides chosen for comparison are all four potential slide areas.

The conditions represented in the five slides also include slide velocities between 37 fps (11.3 m/sec) and 192 fps (58.5 m/sec), and slide volumes between 0.9 and 4.75 million cubic yards (.69 to 3.63 million cubic meters). The ability of the numerical model to reproduce data obtained from the physical model is being tested over a wide range of variables.

In Table 1 the numerical results obtained for the wave height and time of arrival of the first wave crest are compared with the physical model results for the slide condition shown in Figures 7 and 8. The maximum wave height is the most critical item to be obtained from the numerical model. Numerical results are not shown in those cases where the gage location in the physical model falls outside the computational area used in the numerical model. To better illustrate the agreement between the numerical results and the physical model results, the time-wave height history is plotted at six of the more critical gage locations in the reservoir. These results are shown in Figures 9 through 14. The agreement of numerical and physical model results is observed to be excellent at gages 1, 2, 3 and 4. Somewhat less agreement is observed at gages 5 and 6 although the maximum wave height agreement is still satisfactory. The numerical results are compared with the physical model results only until the first wave crest is obtained since the numerical model is not a good representation of the physical phenomena once a significant amount of wave runoff occurs at the reservoir boundaries. In most cases the first wave crest was the largest wave measured in the physical model.

Similar agreement exists between the numerical model and the physical model results for all five slides considered in the numerical investigation. The average difference in the height of the first wave crest was about 25% with an average difference in the times of arrival of the first wave crest of only 9%.

7. CONCLUSIONS. The agreement between the physical model and numerical model results are very satisfactory considering that each set of results is subject to certain inherent limitations imposed by the model theory and others imposed by lack of



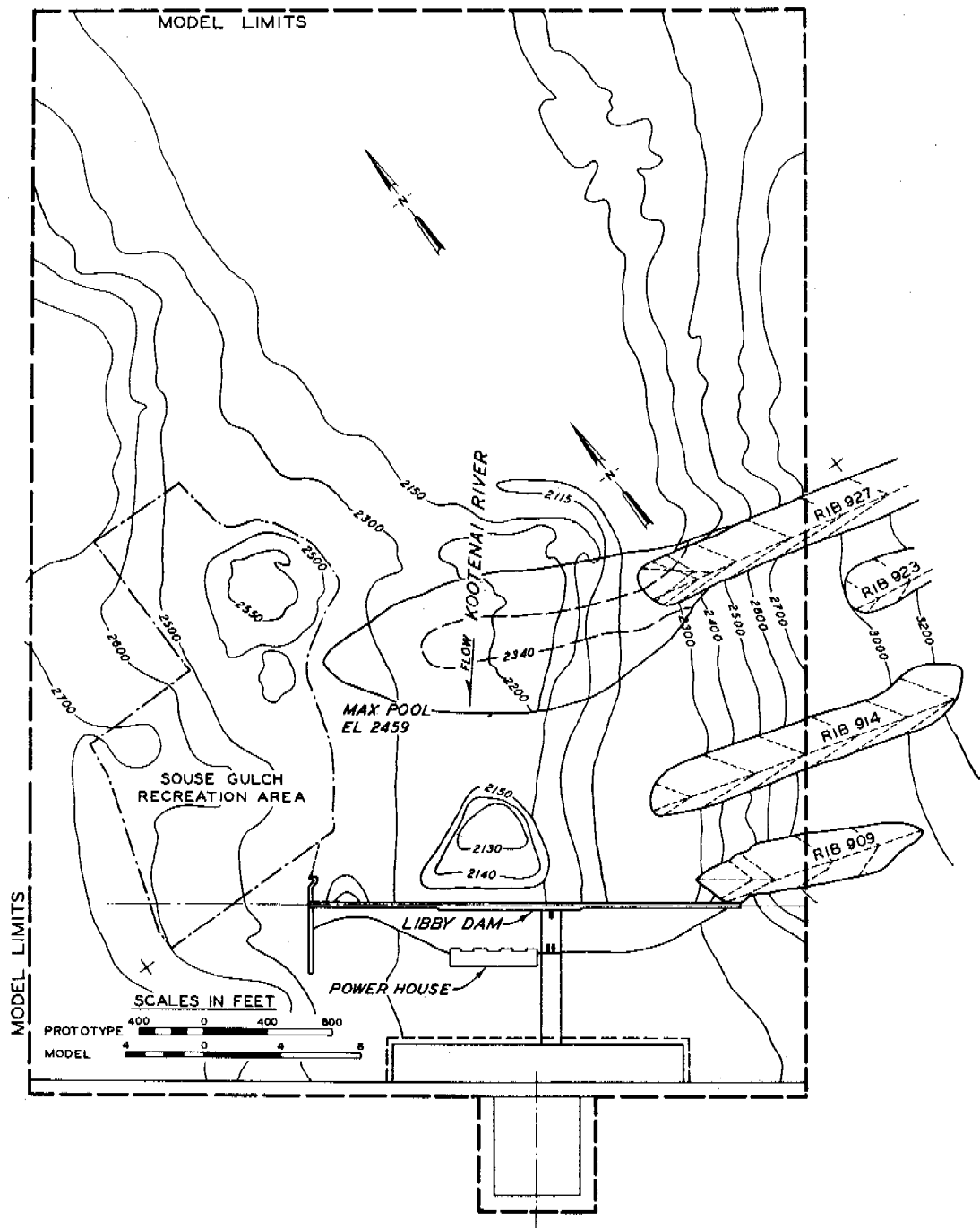


Figure 8. Final position of landslide material for run 87

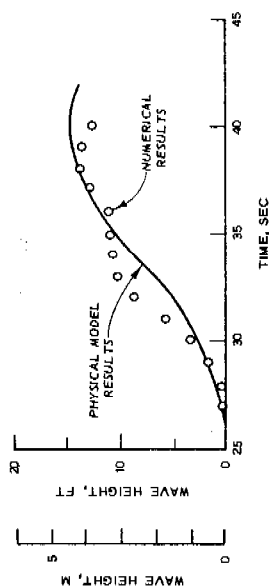


Figure 12. Wave height as a function of time at gage 4 for run 87

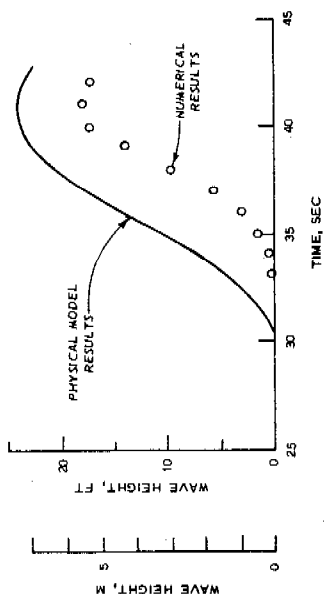


Figure 13. Wave height as a function of time at gage 5 for run 87

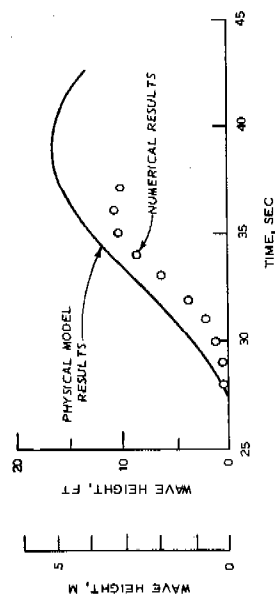


Figure 14. Wave height as a function of time at gage 6 for run 87

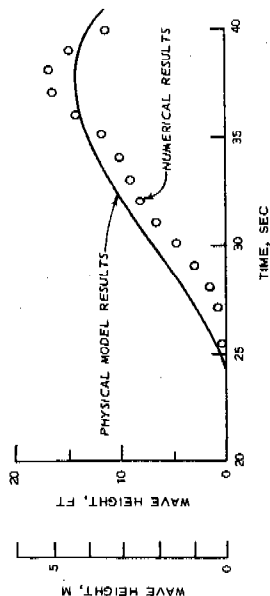


Figure 9. Wave height as a function of time at gage 1 for run 87

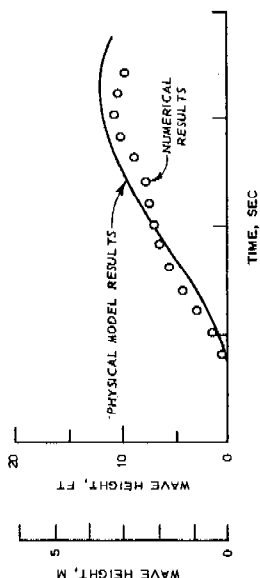


Figure 10. Wave height as a function of time at gage 2 for run 87

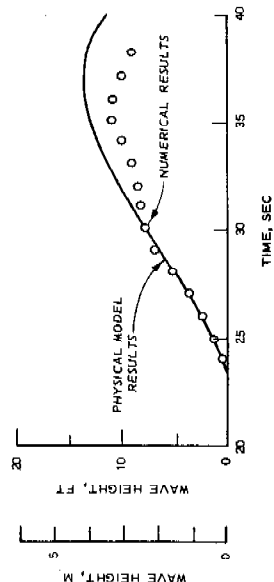


Figure 11. Wave height as a function of time at gage 3 for run 87

knowledge concerning prototype conditions or how to model certain phenomena. The results indicate that the numerical model is capable of modeling landslide generated water waves sufficiently accurate to allow overall engineering decisions to be made concerning the possible effects of a potential landslide.

For small landslide velocities the viscous drag and pressure drag contributions to wave heights and velocities are small and the physical displacement of the water by the landslide is the predominate factor. The water waves produced by the slide will generally be propagating faster than the slide is moving and thus the initial phase of the landslide must be accurately defined if good numerical results are to be obtained.

For large landslide velocities significant contributions to wave heights and propagation velocities are produced by the viscous drag and pressure drag. Viscous drag and pressure drag contributions to wave heights and velocities are focused in the direction of the landslide to a greater degree than the contributions from the physical displacement of water. If the landslide is moving faster than the normal propagation velocity for the water waves it produces, the entire path and time history of the landslide becomes of importance in obtaining an accurate prediction of the first wave crest.

Additional experimental work is needed in which the landslide parameters, which are required as input to the numerical program, are observed and measured in greater detail. The initial physical model study of Libby Dam and Lake Koocanusa was an end in itself and not designed to provide information to verify a numerical model. Fortunately, most of the required information was observed but not in the detail that would be desirable for use in detailed verification of a numerical model.

8. ACKNOWLEDGEMENTS. The work upon which this paper is based was performed while the first writer was working as a Research Engineer with the U.S. Army Engineer Waterways Experiment Station (WES), Vicksburg, Mississippi under an Inter-governmental Personnel Exchange agreement between WES and The University of Alabama. Funds for this investigation were authorized by the Office, Chief of Engineers under the Civil Works Research Program.

TABLE 1

RUN 87 - RIB 927

MAXIMUM SLIDE VELOCITY = 84 fps (25.6 m/sec)

<u>Gage Location</u>	<u>First Wave Crest, Ft (m)</u>		<u>Arrival Time of First Crest, sec.</u>	
	<u>Physical Model</u>	<u>Numerical Model</u>	<u>Physical Model</u>	<u>Numerical Model</u>
1	14 (4.27)	16.9 (5.15)	38	37.5
2	12 (3.66)	10.7 (3.26)	36	35.5
3	14 (4.27)	11.6 (3.54)	37	35.5
4	15 (4.57)	15.3 (4.66)	40	38.5
5	24 (7.32)	18.0 (5.49)	41	41.0
6	17 (5.18)	10.7 (3.26)	39	36.0
7	25 (7.62)	25.1 (7.65)	37	42.5
8	12 (3.66)	11.1 (3.38)	32	32.0
9	17 (5.18)	11.6 (3.54)	29	30.0
10	7 (2.13)	5.7 (1.74)	23	20.0
11	15 (4.57)	19.2 (5.85)	19	23.5
12	12 (3.66)	8.3 (2.53)	19	18.0
13	6 (1.83)	10.1 (3.08)	22	21.5
14	5 (1.52)	----	37	----
15	--	----	--	----
16	--	----	--	----

Average difference in wave  
heights between numerical  
and experimental values 23%

Average difference in time  
of arrival between numerical  
and experimental values 6.2%

## REFERENCES

1. Davidson, D. D. and R. W. Whalin. "Potential Land-slide-Generated Water Waves, Libby Dam and Lake Koocanusa, Montana", Technical Report H-75-15, U. S. Army Engineer Waterways Experiment Station, Vicksburg, Mississippi, December 1974.
2. Hansen, W. "Hydrodynamical Methods Applied to Oceanographic Problems", Proceedings of the Symposium on Mathematical-Hydrodynamical Methods of Physical Oceanography, Institut fur Meereskunde der Universitat, Hamburg 1962.
3. Hwang, Li-San, H. Lee Butler, and D. J. Divoky. "Tsunami Model: Generation and Open-Sea Characteristics", Bulletin of the Seismological Society of America, Vol. 62, No. 6, December 1972.
4. Leendertse, J. J. "Aspects of a Computational Model for Long-Period Water-Wave Propagation", Memorandum Rm-5294-PR, The Rand Corporation, Santa Monica, California, May 1967.
5. Noda, E.K. "Theory of Water Waves Generated by a Time Dependent Boundary Displacement", Hydraulic Engineering Laboratory Report HEL 16-5, University of California (Berkeley), October 1969.
6. Platzman, G. W. "A Numerical Computation of the Surge of 26 June 1954 on Lake Michigan", Geophysics, Vol. 6, No. 3-4, 1959, pp. 407-38.
7. Whalin, Robert W. "The Linear Theory of Water Waves Generated by Explosions", National Engineering Science Co. Report S-256-1, Pasadena, California, October 1965.
8. Wiegel, R. L., E. K. Noda, E. M. Kuba, D. M. Gee and G. F. Thornberg. "Water Waves Generated by Landslides in Reservoirs", Hydraulic Engineering Laboratory Report HEL 19-1, University of California (Berkeley), April 1968.

# NOTATION

$A_c$	= grid cell area
$A_z$	= cross-sectional area of landslide
$C$	= Chezy coefficient
$C_D$	= pressure drag coefficient
$g$	= acceleration of gravity
$h$	= depth of the undisturbed water surface
$L_x, L_y$	= x and y component of the acceleration effect of the landslide on the water
$j, k$	= indices for finite difference grid locations
$n$	= indices indicating multiples of the time step
$\bar{n}$	= unit vector normal to boundary
$R_x, R_y$	= x and y component of the bottom roughness effect
$t$	= time
$u$	= depth-averaged water velocity component in the x direction
$v$	= depth-averaged water velocity component in the y direction
$\bar{v}$	= velocity vector with components u and v
$V_x$	= x component of the landslide velocity
$V_y$	= y component of the landslide velocity
$w$	= water velocity in the z direction
$x, y$	= rectangular coordinate variables
$\beta$	= pressure drag parameter
$\eta$	= water level displacement with respect to still water elevation
$\zeta$	= vertical bottom deformation created by landslide material
$\rho$	= density of water

# AUTOMATIC EULER-MACLAURIN INTEGRATION

Julia H. Gray

and

L. B. Rall

Mathematics Research Center  
University of Wisconsin-Madison  
Madison, Wisconsin 53706

ABSTRACT. The Euler-Maclaurin formula for numerical integration is

$$\int_a^b f(x)dx = T_n - \sum_{m=2}^{2k-1} \frac{h^m B_m}{m!} [f^{(m-1)}(b) - f^{(m-1)}(a)] - \frac{h^{2k}}{(2k)!} (b-a) B_{2k} f^{(2k)}(\xi) ,$$

where  $T_n$  is the trapezoidal rule

$$T_n = \frac{h}{2} [f(a) + f(b)] + h \sum_{i=1}^{n-1} f(a+ih) ,$$

$h = (b-a)/n$ ,  $B_2, B_3, \dots, B_{2k}$  are Bernoulli numbers, and  $a < \xi < b$ . The application of this formula can be automated by using software developed at MRC for analytic differentiation and interval analysis. The use of interval techniques permits rigorous bounding of the error due to roundoff, and also the truncation error by calculating an interval containing  $f^{(2k)}(\xi)$ . By use of observed values of the time required for evaluation of the integrand and experimental results on differentiation time, optimal values for  $n, k$  are calculated to give a required accuracy in minimum time, or an estimate of the ultimate accuracy of the Euler-Maclaurin integration method may be computed. The theory is illustrated by results obtained using a UNIVAC 1108/1110 program.

AMS (MOS) Classifications (1970): 65D30, 65G05.

Key Words: Numerical integration, Euler-Maclaurin formulas, Automatic error estimation, Interval integrals.

---

Research sponsored by the U. S. Army under Contract No. DAAG29-75-C-0024.

1. BACKGROUND. With the aid of software for interval analysis [4] and automatic differentiation, various methods for numerical integration with rigorous error estimation [1, 3] have been implemented as a program for the UNIVAC 1108/1110 [2]. To be more specific, the program described in [2] provides the user with the capability of finding intervals containing the value of the integral

$$(1.1) \quad z = \int_a^b f(x) dx$$

by the use of Riemann sums, various open and closed Newton-Cotes integration formulas, or standard Gaussian integration formulas. The latter types of integration formulas (Newton-Cotes and Gaussian) are of the form

$$(1.2) \quad z = r(f) + e(f) ,$$

where the rule  $r(f)$  of numerical integration is a linear combination

$$(1.3) \quad r(f) = \sum_{i=1}^n f(x_i) w_i$$

of values of the integrand at the nodes  $x_1, x_2, \dots, x_n$ , with weights  $w_1, w_2, \dots, w_n$ , and the (truncation) error (or remainder) term

$$(1.4) \quad e(f) = c_n(a, b) \frac{f^{(k)}(\xi)}{k!} , \quad a < \xi < b ,$$

is a multiple of the  $k$ th Taylor coefficient of  $f$ , evaluated at some point  $\xi$  in the open interval  $(a, b)$ . In (1.4), the constant  $c_n(a, b)$  is independent of  $f$ . A numerical integration formula of this type will be said to be of order  $n$  and degree  $k$ , and is valid for integrands which are sufficiently smooth. As a typical example, one has the (extended) trapezoidal formula [5, p. 170]

$$(1.5) \quad \int_a^b f(x) dx = \frac{b-a}{n} \left[ \frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f\left(a+i\left(\frac{b-a}{n}\right)\right) \right] - \frac{(b-a)^3}{6n^2} \cdot \frac{f''(\xi)}{2!} , \quad a < \xi < b ,$$

which is of order  $n+1$ ,  $n$  a positive integer, and degree two.

As the program has facilities for analytic differentiation, it is also possible to implement formulas for numerical integration in which the rule involves values of derivatives (or, equivalently, Taylor coefficients) of the integrand. A class of formulas of this type may be obtained from the Euler-Maclaurin formula for numerical



integration, in which the order ( $\geq 2$ ) and (even) degree may be specified by the user. This formula and its implementation for automatic computation are described in the following sections.

As in [2], the present program is applicable only to integrands  $f(x)$  which may be written in ordinary FORTRAN notation. In addition to the variable of integration, the integrand may contain one or more parameters to be specified by the user.

2. THE EULER-MACLAURIN FORMULA. Using the fact that the Bernoulli numbers  $B_3, B_5, B_7, \dots$  of odd order  $> 3$  all vanish [5, p. 218], the Euler-Maclaurin integration formula of order  $n+1$  and degree  $2k$  may be written in terms of values of the integrand and its Taylor coefficients as

$$(2.1) \quad \int_a^b f(x)dx = h \left[ \frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(a+ih) \right] - \\ - \sum_{m=1}^{k-1} \frac{h^{2m} B_{2m}}{2m} \left[ \frac{f^{(2m-1)}(b)}{(2m-1)!} - \frac{f^{(2m-1)}(a)}{(2m-1)!} \right] - \\ - h^{2k} (b-a) B_{2k} \frac{f^{(2k)}(\xi)}{(2k)!}, \quad a < \xi < b,$$

where

$$(2.2) \quad h = \frac{b-a}{n}.$$

It should be noted that the values of  $\xi$  for which formula (2.1) holds depend in general on  $a, b, f, k$ , and  $n$ ; however, this dependence is suppressed for simplicity of notation, and will be inconsequential in the interval version of the formula.

For practical reasons, the computer program is limited to calculation of Taylor coefficients of orders less than twenty, hence the maximum degree of the integration formula (2.1) permitted without modification of the program is  $2k = 18$ . The even-order Bernoulli numbers  $B_2, B_4, \dots, B_{18}$  are given in Table 2.1 [5, p. 218].

$B_2 = \frac{1}{6} ,$	$B_4 = -\frac{1}{30} ,$	$B_6 = \frac{1}{42} ,$
$B_8 = -\frac{1}{30} ,$	$B_{10} = \frac{5}{66} ,$	$B_{12} = -\frac{691}{2730} ,$
$B_{14} = \frac{7}{6} ,$	$B_{16} = -\frac{3617}{510} ,$	$B_{18} = \frac{43867}{798} .$

TABLE 2.1. THE BERNOULLI NUMBERS

$B_2, B_4, \dots, B_{18} .$

The first term of (2.1),

$$(2.3) \quad t_n = t_n(f) = h \left[ \frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(a + ih) \right]$$

is simply the trapezoidal rule from the integration formula (1.5). Hence, the use in the second term of (2.1) of values of the Taylor coefficients of  $f(x)$  at the endpoints  $a, b$  can be viewed as a method for increasing the order of accuracy of the trapezoidal integration formula.

3. THE INTERVAL VERSION OF THE EULER-MACLAURIN FORMULA. In order to make use of the integration formula (2.1) and take into account the effects of round-off error, uncertainties in the coefficients of  $f(x)$  and perhaps the limits of integration, and the unknown value of  $\xi$ , an interval extension of the right-hand side of (2.1) is calculated by the program. As defined in [2], an interval extension of a set of numbers is any interval containing that set. (Of course, the set being extended could consist of a single number.) Interval extensions of numbers and functions will generally be denoted by the corresponding capital letters, exceptions being small integers and the Bernoulli numbers. Taking

$$(3.1) \quad X \supset [a, b]$$

and

$$(3.2) \quad H = \frac{B-A}{n} ,$$

one computes the interval version

$$\begin{aligned}
 (3.3) \quad I_{n,k} = & H \left[ \frac{F(A) + F(B)}{2} + \sum_{i=1}^{n-1} F(A + iH) \right] - \\
 & - \sum_{m=1}^{k-1} \frac{H^{2m} B_{2m}}{2m} \left[ \frac{F^{(2m-1)}(B)}{(2m-1)!} - \frac{F^{(2m-1)}(A)}{(2m-1)!} \right] - \\
 & - H^{2k} (B-A) B_{2k} \frac{F^{(2k)}(X)}{(2k)!}
 \end{aligned}$$

of the Euler-Maclaurin integration formula.

For

$$(3.4) \quad z = \int_a^b f(x) dx,$$

one has that

$$(3.5) \quad z \in I_{n,k}$$

for all positive integers  $k, n$ . If  $I_{n,k} = [c, d]$ , then one may take the midpoint

$$(3.6) \quad z^* = \mu[I_{n,k}] = \frac{c+d}{2}$$

as an approximation to  $z$ , with absolute error

$$(3.7) \quad |z - z^*| \leq \varepsilon = \frac{1}{2} \delta[I_{n,k}] = \frac{d-c}{2}.$$

If the minimum bound for the relative (or percentage) error is desired instead, then, provided  $0 \notin I_{n,k}$ , the corresponding estimate for  $z$  is the harmonic point

$$(3.8) \quad z^{**} = \eta[I_{n,k}] = \frac{2cd}{c+d},$$

with relative error

$$(3.9) \quad \left| \frac{z - z^{**}}{z} \right| \leq \rho = \left| \frac{d - c}{c + d} \right|.$$

The percentage error is, of course, bounded by  $100\rho$ . The a posteriori error bounds (3.7) and (3.9) are rigorous, as follows from the theory of interval analysis [3, 4].

4. THE INTERSECTION PRINCIPLE. As the computer program calculates Taylor coefficients recursively [2], the values of  $I_{n,1}, I_{n,2}, \dots, I_{n,k-1}$  can be obtained in the course of the computation of  $I_{n,k}$  with very little additional effort. For

$$(4.1) \quad I = \bigcap_{j=1}^k I_{n,j},$$

it follows from (3.5) that

$$(4.2) \quad z = \int_a^b f(x) \in I.$$

The estimates

$$(4.3) \quad z^* = \mu[I], \quad \epsilon = \frac{1}{2} \delta[I],$$

or, if  $0 \notin I$ ,

$$(4.4) \quad z^{**} = \eta[I], \quad \rho = 2 \left| \frac{\delta[I]}{\mu[I]} \right|,$$

are at least as good as those obtained from (3.6)-(3.9), and are the ones actually calculated by the program. This is a simple application of the intersection principle of interval analysis, which states that results belonging to several intervals are contained in their intersection.

For example, for

$$(4.5) \quad z = \int_0^1 \frac{\sin x + \tan^{-1} x}{\ln(2 + e^x)} dx ,$$

one has

$$(4.6) \quad z \in I_{1,1} = [0.43180767, 1.0839293] ,$$

the interval form of the trapezoidal formula, and

$$(4.7) \quad z \in I_{1,2} = [0.41804086, 0.81363283] ,$$

and thus

$$(4.8) \quad z \in I = I_{1,1} \cap I_{1,2} = [0.43180767, 0.81363283] ,$$

which gives a more accurate result.

5. OPTIMIZATION. If the values of  $n$  and  $k$  are given by the user of the Euler-Maclaurin integration program, then the results given by (4.3) or (4.4) are obtained directly. On the other hand, one or both of these parameters may be determined by the program in order to achieve a prescribed or maximum possible accuracy with the use of the minimal amount of computational effort. This type of optimization of performance of the program makes use of an extension of the strategy outlined previously for  $k$  fixed and  $n$  arbitrary [2, pp. 12-16] to the case that both the order and degree of the integration rule may vary.

First of all, for  $n$  fixed, it has been found expedient simply to compute

$$(5.1) \quad I^{(j)} = \bigcap_{i=1}^j I_{n,i}$$

until

$$(5.2) \quad I^{(j)} = I^{(j+1)} = I^{(j+2)}$$

or  $j = 9$ . Once the optimal value  $j = j^*$  is determined, one sets

$$(5.3) \quad I = I^{(j^*)}$$

and obtains approximate values for the integral and error estimates from (4.3) or (4.4). For example, for the integral (4.5),

$$(5.4) \quad I^{(2)} = [0.43180767, 0.81363283]$$

is optimum for  $n = 1$ , while

$$(5.5) \quad I^{(3)} = [0.65189380, 0.66617152]$$

is optimum for  $n = 2$ , and

$$(5.6) \quad I^{(6)} = [0.65887366, 0.65890718]$$

is the optimal value for  $n = 4$ .

In the second case of fixed  $k$  and arbitrary  $n$ , the procedure is similar to optimization of integration rules of degree  $k$  done previously [2, pp. 12-16]. In order to discuss the analysis, write

$$(5.7) \quad I_{n,k} = T_n + S_{n,k} + R_{n,k},$$

where

$$(5.8) \quad T_n = H \left[ \frac{F(A) + F(B)}{2} + \sum_{i=1}^{n-1} F(A+iH) \right],$$

the interval version of the trapezoidal rule,

$$(5.9) \quad S_{n,k} = - \sum_{m=1}^{k-1} \frac{H^{2m} B_{2m}}{2m} \left[ \frac{F^{(2m-1)}(B)}{(2m-1)!} - \frac{F^{(2m-1)}(A)}{(2m-1)!} \right],$$

the correction term, and

$$(5.10) \quad R_{n,k} = -H^{2k} (B-A) B_{2k} \frac{F^{(2k)}(X)}{(2k)!},$$

the interval remainder term. The error bound

$$(5.11) \quad \frac{1}{2} \delta[I_{n,k}] = \frac{1}{2} \delta[T_n] + \frac{1}{2} \delta[S_{n,k}] + \frac{1}{2} \delta[R_{n,k}]$$

will thus depend on the way in which the widths of the intervals (5.8)-(5.10) vary with  $n$ , or, more conveniently, as functions of

$$(5.12) \quad h = \frac{\delta[X]}{n}.$$

The quantity  $\delta[T_n]$  turns out to be essentially constant as a function of  $n$ , as  $T_n$  is the average of interval evaluations of the integrand  $f(x)$ , the widths of these intervals depending on round-off and imprecision in the data involved in the definition of  $f(x)$ . Thus,

$$(5.13) \quad r = \delta[T_1]$$

is computed, and is taken as a measure of the potential accuracy of the numerical integration procedure. If an error bound  $\epsilon$  is prescribed such that

$$(5.14) \quad \epsilon \leq \frac{1}{2} r,$$

this accuracy will be considered to be unattainable, and the program will print an error message [2, p. 55]. If an error bound  $\epsilon$  is not specified in advance, then the value of  $r$  is used to obtain an estimate of the maximum number  $p$  of decimal places of accuracy that can be guaranteed for the approximate value of the integral. One takes  $p$  to be the largest positive integer such that

$$(5.15) \quad 10^{-p} > r,$$

and sets

$$(5.16) \quad \epsilon = 5 \cdot 10^{-p-1}$$

as the desired error bound [2, p. 15]. In the program, (5.16) is computed if the input value  $EPS = 0$  is specified.

Define

$$(5.17) \quad r_m = \delta \left[ \frac{B_{2m}}{2m} \left\{ \frac{F^{(2m-1)}(B)}{(2m-1)!} - \frac{F^{(2m-1)}(A)}{(2m-1)!} \right\} \right], \quad m = 1, 2, \dots, k-1,$$

and

$$(5.18) \quad t = \delta \left[ (B-A) B_{2k} \frac{F^{(2k)}(X)}{(2k)!} \right].$$

One has

$$(5.19) \quad \delta[I_{n,k}] \leq p(h) = r + r_1 h^2 + \dots + r_{k-1} h^{2k-2} + t h^{2k},$$

and thus

$$(5.20) \quad \frac{1}{2} \delta[I_{n,k}] < \varepsilon$$

if  $n$  is chosen large enough so that

$$(5.21) \quad p(h) < 2\varepsilon.$$

In order to conserve computer time, the optimal solution is taken to be the smallest positive integer  $n$  for which (5.21) is satisfied. This value is determined by a simple iteration, taking into account the fact that the coefficient  $t$  in (5.19) is ordinarily much larger than the values of  $r, r_1, \dots, r_{k-1}$ . One uses as an initial approximation the least positive integer  $n$  such that

$$(5.22) \quad n > \frac{\delta[X]}{2k \sqrt{\frac{2\varepsilon-r}{t}}},$$

and, if necessary, increases  $n$  by one until (5.19) is satisfied by the corresponding value of  $h$ .



In the final case, the values of both  $n$  and  $k$  are to be determined in order to satisfy the error bound  $\varepsilon$  with minimal computational effort. To do this, some assumptions are made concerning the dependence upon  $k$  of the width

$$(5.23) \quad t_k = \delta \left[ \frac{F^{(k)}(X)}{k!} \right]$$

of the  $k$ th interval Taylor coefficient of the integrand  $f(x)$ , and the time

$$(5.24) \quad \theta_k = \theta \left[ \frac{F^{(k)}(X)}{k!} \right]$$

required for its computation. Knowing these relationships, one could determine if a given accuracy is best obtained by increasing  $n$  or  $k$ .

A heuristic argument indicates that the values of  $t_k$  are given by the sums of an arithmetic progression, while the  $\theta_k$  are obtained in the same way from a geometric progression. These observations are borne out experimentally, as illustrated in Table 5.1 for

$$(5.25) \quad f(x) = \frac{\sin x + \tan^{-1} x}{\ln(2 + e^x)}, \quad X = [0, 1].$$

The program computes

$$(5.26) \quad \theta_0 = \theta[T_1] = \theta \left[ \frac{F(A) + F(B)}{2} \right],$$

from which the estimate

$$(5.27) \quad \theta_n = \left( \frac{n+1}{2} \right) \theta_0$$

is obtained for  $\theta[T_n]$ . Similarly, the values of  $\theta \left[ \frac{F^{(k)}(X)}{k!} \right]$  and  $\delta \left[ \frac{F^{(k)}(X)}{k!} \right]$  are obtained initially for  $k = 0, 1, 2$ . Suppose that  $n_k$  denotes the least value of  $n$  satisfying inequality (5.22). The corresponding estimate for  $\theta[I_{n_k, k}]$  is taken to be

k	$\theta \left[ \frac{F^{(k)}(X)}{k!} \right]$	$\delta \left[ \frac{F^{(k)}(X)}{k!} \right]$
0	0.130 sec.	1.4809
1	0.259 "	2.0949
2	0.397 "	3.9133
3	0.546 "	6.5944
4	0.701 "	$1.1868 \times 10$
5	0.865 "	$2.1230 \times 10$
6	1.038 "	$3.8380 \times 10$
7	1.220 "	$7.0895 \times 10$
8	1.409 "	$2.6012 \times 10^2$
9	1.609 "	$2.6012 \times 10^2$
10	1.817 "	$5.1869 \times 10^2$
11	2.034 "	$1.0605 \times 10^3$
12	2.261 "	$2.2173 \times 10^3$
13	2.498 "	$4.7251 \times 10^3$
14	2.745 "	$1.0231 \times 10^4$
15	3.000 "	$2.2448 \times 10^4$
16	3.266 "	$4.9771 \times 10^4$
17	3.541 "	$1.1131 \times 10^5$
18	3.828 "	$2.5068 \times 10^5$
19	4.124 "	$5.6780 \times 10^5$

TABLE 5. 1. COMPUTATION TIMES AND WIDTHS  
OF INTERVAL TAYLOR COEFFICIENTS OF (5.25).

$$(5.28) \quad \theta_{n_k, k} = \left( \frac{n_k + 1}{2} \right) \theta_0 ,$$

for the additional time to compute  $T_{n_k}$ . Using the numbers

$$(5.29) \quad \bar{t} = B_{2k+2} \delta_{2k}^3 / \delta_{2k-1}^2 ,$$

where

$$(5.30) \quad \delta_k = \delta \left[ \frac{F^{(k)}(X)}{k!} \right]$$

an estimate  $\bar{n}_{k+1}$  for  $n_{k+1}$  can also be obtained by using the value of  $\bar{t}$  in (5.22) in place of  $t$ . Similarly, one estimates

$$(5.31) \quad \bar{\theta}_{k+1} = 3\theta \left[ \frac{F^{(2k)}(X)}{(2k)!} \right] - 2\theta \left[ \frac{F^{(2k-1)}(X)}{(2k-1)!} \right] .$$

Now, if

$$(5.32) \quad \theta_{n_k, k} < \bar{\theta}_{k+1} + \left( \frac{\bar{n}_{k+1} + 1}{2} \right) \theta_0 ,$$

the program computes

$$(5.33) \quad I = \bigcap_{j=1}^{2k} I_{n_k, j} ;$$

otherwise, the above process is continued with  $k$  replaced by  $k + 1$  until (5.32) is satisfied or  $k = 9$ .

## REFERENCES

1. Gray, Julia H. and L. B. Rall. A computational system for numerical integration with rigorous error estimation, Proceedings of the 1974 Army Numerical Analysis Conference, pp. 341-355. ARO Report 74-2, U. S. Army Research Office, Research Triangle Park, N. C., 1974.
2. Gray, Julia H. and L. B. Rall. INTE: A UNIVAC 1108/1110 Program for numerical integration with rigorous error estimation, MRC Technical Summary Report #1428, Mathematics Research Center, University of Wisconsin-Madison, 1975.
3. Moore, R. E. The automatic analysis and control of error in digital computation based on the use of interval numbers, Error in Digital Computation, Vol. I, ed. by L. B. Rall, pp. 61-130. John Wiley & Sons, New York, 1965.
4. Moore, R. E. Interval analysis, Prentice-Hall, Englewood Cliffs, N. J., 1966.
5. Mysovskih, I. P. Lectures on numerical methods, Tr. from Russian by L. B. Rall, Wolters-Noordhoff, Gronigen, The Netherlands, 1969.

UNIVERSITY OF WISCONSIN - MADISON  
MATHEMATICS RESEARCH CENTER

CANCELLATION AND ROUNDING ERRORS

J. Barkley Rosser and J. Michael Yohe

Technical Summary Report #1588

ABSTRACT

The fact that a calculating machine can usually hold only an approximation to the number that one is concerned with leads to cancellation errors and rounding errors. These concepts are defined precisely and examples are given. Suggestions are given for reducing (when possible) the size of errors arising from these two effects.

AMS(MOS) Subject Classifications - 68-01, 68A05, 68A10

Key Words - Computer arithmetic, Rounding errors, Cancellation, Numerical algorithms.

## CANCELLATION AND ROUNDING ERRORS

J. Barkley Rosser and J. Michael Yohe

1. Definition of terms. We assume that all real numbers with which we will be dealing are represented on the calculating machine in floating point positional notation with a fixed radix. If two such numbers are nearly equal, then the calculated difference between them will have more leading zeros than either of the original numbers. This will usually cause no trouble if it happens that the numbers are exactly representable on the calculating machine, though even in this case imperfections in machine architecture can result in a botched answer. However, in the usual case, when both numbers are only approximated on the calculating machine, this phenomenon will commonly cause appreciable relative error in the answer. As most real numbers can only be approximated on a calculating machine, this danger is always present, and steps should be taken to minimize the effect.

Suppose we are given two real numbers  $a$  and  $b$  and a binary operation  $\circ$ , and suppose also that we have a machine  $M$  with the corresponding binary operation  $\circ_M$ . Suppose further that we have a rounding operation  $\rho$  from the real numbers to the set of machine numbers; we shall assume that  $\rho$  rounds a real number to its closest approximation on the machine.

---

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

We are interested in how well the machine answer,  $\rho(a) \circ_M \rho(b)$ , will approximate the true answer,  $a \circ b$ : that is, we wish information on the value of

$$(1.1) \quad |\rho(a) \circ_M \rho(b) - a \circ b|.$$

There are actually three different considerations involved in this expression; we shall call them machine error (or error due to architectural deficiencies in the machine), accumulated error (which is due to performing arithmetic operations on approximate numbers), and rounding error (which results only from the fact that most real numbers must be approximated). We shall now define these types of error precisely.

Machine error is the amount by which the machine result fails to be the best possible result derivable from the rounded numbers being given, including the fact that the result must be rounded; i.e., machine error is defined to be

$$(1.2) \quad |\rho(a) \circ_M \rho(b) - \rho(\rho(a) \circ \rho(b))|.$$

In a well-designed machine, (1.2) can be made to vanish; see [9] for details. Thus this type of error is completely avoidable. Unfortunately, it is often not avoided; we have even seen a calculating machine on which, for a large class of real numbers,  $a$ , we have  $a \times_M 1 - a \times 1 \neq 0$  in spite of the fact that  $\rho(a) = a$  for this class!

Accumulated error is the amount by which the rounded result of performing the real operation using rounded operands differs from the rounded result using the original operands; i. e. ,

$$(1.3) \quad |\rho(\rho(a) \circ \rho(b)) - \rho(a \circ b)| .$$

In general, accumulated error can not be avoided, but it can be controlled to a certain extent by selecting computational formulas judiciously, as we shall see in the sequel.

Rounding error is simply the amount by which the rounded result differs from the true result, i. e. ,

$$(1.4) \quad |\rho(a \circ b) - a \circ b| .$$

Rounding error must be regarded as totally unavoidable in the context in which we are working.

We observe that

$$(1.5) \quad \begin{aligned} & |\rho(a) \circ_M \rho(b) - a \circ b| \leq \\ & |\rho(a) \circ_M \rho(b) - \rho(\rho(a) \circ \rho(b))| + \\ & + |\rho(\rho(a) \circ \rho(b)) - \rho(a \circ b)| + \\ & + |\rho(a \circ b) - a \circ b| . \end{aligned}$$



Hence, by (1.2) - (1.4), the total absolute error is not greater than the sum of machine error, accumulated error, and rounding error.

Since machine error is avoidable, we shall not treat it here, even though it will be exhibited in some of our discussions: we shall concentrate on accumulated error and rounding error. These can be lumped together and called "error of approximation"; a fuller discussion of approximation generally can be found in Yohe [10, p. 10]. However, since rounding error is totally unavoidable, the major thrust of this paper will be in dealing with accumulated error.

In most cases the relative error associated with an instance of accumulated error behaves rather predictably. That is, the relative error of a result in a case of accumulated error is related in a rather straightforward manner to the relative errors in the operands. In one case, however, the relative error can grow catastrophically; the subtraction of two nearly equal quantities can cause the relative error to grow by many orders of magnitude. This particular instance of accumulated error is so serious that we give it a special name: cancellation error.

As an example, consider

$$(1.6) \quad \frac{22}{7} - \pi .$$

We have

$$\frac{22}{7} \cong 3.142857142857$$

$$\pi \cong 3.141592653590$$

$$\frac{22}{7} - \pi \cong 0.00126448926735.$$

However, suppose the calculation is done on the 8-place decimal machine belonging to one of the authors. We enter the approximations

$$(1.7) \quad \frac{22}{7} \cong 3.1428571$$

$$(1.8) \quad \pi \cong 3.1415927.$$

Subtracting these gives

$$(1.9) \quad \frac{22}{7} - \pi \cong 0.0012644.$$

The value of (1.6) is calculated by the machine to only five significant decimals, of which the last is incorrectly rounded. Thus we have lost at least three decimal places of accuracy.

The cause of cancellation error is the fact that when the calculating machine is called upon to subtract two nearly equal numbers, the early digits cancel, and fewer significant digits will result, as shown.

Rounding error usually occurs as soon as a number is entered into the calculating machine. Thus, in (1.7), the value given is too small, though it is the best possible on an 8-place machine. In (1.8), the value given is too large, though it is the best possible on an 8-place machine. When one performs arithmetic operations on such

approximate numbers, the errors already present can accumulate.

Thus, in (1.9), not only are we giving only five significant digits (due to cancellation error), but we are not even giving the five best digits.

A more accurate five digit value would be

$$\frac{22}{7} - \pi \cong 0.0012645 .$$

The additional discrepancy (getting 0.0012644 instead of 0.0012645) was due to accumulation of rounding errors.

Although we will not discuss it in this report, we should recognize yet one other source of error in using a calculating machine. This is numerical instability, which arises when the method of computation is such that the percent error at one step is multiplied by a constant greater than unity in making the next step.

If one can use double or triple precision arithmetic, so that one has a superfluity of digits, losing a few digits by cancellation errors or rounding errors will likely do no harm. However, one should give thought to the details of the calculation, since occasions can arise in which double precision (or even triple precision !) will not suffice to avoid serious errors, or even complete nonsense.

2. Cancellation errors. As noted, cancellation errors arise when one subtracts two nearly equal numbers. Addition of positive numbers causes no appreciable cancellation errors. Likewise for addition of negative numbers. Also, the way most calculating machines are built, multiplying or dividing two numbers produces no appreciable cancellation error .

Many times when one has a formula involving subtraction that leads to cancellation errors, one can find a mathematically equivalent formula for which there is no cancellation error. A classic illustration arises in solving quadratic equations. Suppose one wishes both roots of

$$(2.1) \quad x^2 - 200x + 1 = 0.$$

The formula for the roots is

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

With the coefficients shown, this gives

$$\frac{200 \pm \sqrt{40000 - 4}}{2}.$$

We have

$$\sqrt{40000 - 4} \cong 199.9899997499875.$$

To get the larger root, we enter 200 into the machine, and the best possible eight digit approximation for the square root, namely 199.99000. When we add and divide by 2, we get 199.99500, which is correct to eight digits. When we subtract and divide by 2, we get

$$0.0050000000.$$

To eight places, the answer should be

$$0.0050001250.$$

Here we have a cancellation error.

In this case, one can find a mathematically equivalent formula for the roots by multiplying both top and bottom of the quadratic formula by

$$-b \mp \sqrt{b^2 - 4ac} .$$

This gives

$$\frac{2c}{-b \mp \sqrt{b^2 - 4ac}} .$$

Then for the small root, we have

$$\frac{2}{200 + 199.99000} \cong 0.0050001250 .$$

As a matter of fact, the situation could have been considerably worse. Suppose we had tried to get the roots of

$$x^2 - 100x + 1 = 0 .$$

This gives the roots

$$\frac{100 \pm \sqrt{10000 - 4}}{2} .$$

We have

$$\sqrt{10000 - 4} \cong 99.9799979996 .$$

We enter 100 into the machine and the best possible eight digit approximation for the square root, namely 99.979998. As the calculator is an eight digit machine, it must reduce these to the same number of decimals before adding or subtracting. It shows 100 as 100.00000; to get the other number to match, it truncates the final 8, so that it gives

$$\frac{100 + 99.97999}{2} = 99.989995 .$$

To eight places the larger root should be approximated by 99.989999.

This is not a cancellation error. This is poor machine architecture.

We should note that some very large and very expensive calculating machines make the same sorts of blunders. For instance, the UNIVAC 1108 and 1110 do so. Other vagaries of calculating machines are discussed in Kahan [6]. A discussion of how to improve calculating machines in matters of this sort is given in Yohe [9].

It is fairly common that, with a little ingenuity, one can find for a formula that produces cancellation error another, mathematically equivalent, formula that does not. Thus, in Rosser [1], we had the formula numbered (2.44)

$$c_{n+1} = \frac{1}{2}(a_n - b_n),$$

which in Table 2.6 did not even give the right order of magnitude for  $c_{11}$ . However, the formula turns out to be mathematically equivalent to the formula numbered (2.88)

$$c_{n+1} = \frac{c_n^2}{4a_{n+1}},$$

which clearly has no cancellation error. Or in the same report, the formula numbered (2.95) involved a difference, and gave disastrous results; the mathematically equivalent formula numbered (2.98) involved

a sum, and gave quite satisfactory results. Of course, just having an algebraic sum won't guarantee avoidance of cancellation error; one summand could be nearly the negative of the other. However, the formula numbered (2.98) involved a sum of squares, and hence could be relied on not to give a cancellation error.

So, for purposes of calculation, close attention should be paid to trying to replace formulas that could produce cancellation errors by mathematically equivalent formulas which do not. Sometimes, this does not seem possible. Thus in the formula numbered (3.27) in Rosser and Papamichael [ 2 ], we are undertaking to compute the  $A_N$  recursively from

$$(2.2) \quad A_{N+1} = \frac{1}{E_1} \left\{ B_0^{\frac{1}{2}} D_N - \sum_{r=2}^{N+1} E_r A_{N+2-r}^{(2r-1)} \right\} .$$

The  $E_r$  are known, with  $E_1 \approx 2.4$ . The value  $(-1)^N B_0^{\frac{1}{2}} D_N$  is approximately 1.5. The  $A_{N+2-r}^{(2r-1)}$  can be calculated from  $A_1, \dots, A_N$ . Now the  $A_N$  decrease in absolute value;  $A_{20} \approx -10^{-8}$ . So obviously use of the formula shown will have to involve serious cancellation errors for the larger values of  $N$ . However, we were not able to find any other way to calculate that  $A_N$ . By using double precision, we were able to get about two significant digits for  $A_{20}$ . To have gone appreciably further would have required a triple precision calculation.

Incidentally, this also disclosed a clearly marked case of numerical instability. If one works out the coefficient of  $A_N$  on the right side of (2.2), it is of the order of 3. In other words, whatever absolute error we make in  $A_N$ , we are guaranteed to make an absolute error about 3 times as large in  $A_{N+1}$ .

3. Rounding errors. As we indicated, rounding errors originate from the fact that usually a calculator can hold only an approximation to a given number. When operations are performed on these approximate numbers, the errors tend to accumulate. A classic treatise on this subject is Wilkinson [3].

Roughly speaking, if one multiplies or divides two numbers, the percent error in the product or quotient can be as large as the sum of the absolute percent errors in each of the participating numbers. It need not be that large, and often will not be. When it comes to adding or subtracting two numbers, the situation is much more complex. In subtracting, the absolute error of the difference might be expected not to be more than the sum of the absolute errors of the participating terms (this is not always so) but if there is cancellation error, the percent error could make a sensational jump.

Thus, in our early example, we had an approximation for  $22/7$  with about  $(1.4 \times 10^{-6})\%$  error and an approximation for  $\pi$  with about  $(1.5 \times 10^{-6})\%$  error, but got an approximation for  $(22/7) - \pi$  with about  $(7.1 \times 10^{-3})\%$  error.



Even in addition of positive numbers, the situation is murky.

It might seem obvious that if one has approximations  $a_1, \dots, a_N$  on the machine, each of which is not more than  $\epsilon$  percent less than the true value, then the sum  $a_1 + a_2 + \dots + a_N$  will not be more than  $\epsilon$  percent less than the sum of the true values. This seems merely to be a special case of the distributive law of algebra. However, it is not so. See page 11 of Yohe [10].

Consider the divergent series

$$(3.1) \quad 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots$$

Let us propose to sum this (or parts of it) on an eight decimal digit calculating machine which can accept or handle numbers down to an underflow limit of about  $10^{-100}$ . Each term of the series (down to the underflow limit) can be entered on the calculator to within  $\epsilon$  percent accuracy. Suppose we do this successively, and add. With an 8-place decimal calculating machine, after fewer than  $10^8$  terms the terms will be so small that when they are added they will make no change in the calculated sum. From that point on the calculated sum will remain the same, though the true sum will rise gradually into the hundreds until we come to terms so small that underflow limitations prevent us from entering them into the machine, and the summation is perforce terminated.

Of course, this is a very contrived situation, but it illustrates why it is difficult to make any general remarks about rounding errors for addition, and even less for subtraction. However, in general, a group

of numbers to be added will not be too disparate in size, and something like the distributive law can be invoked, but with caution.

Usually, with ideal rounding, the errors from rounding are reasonably random. This tends to hold down the accumulation of errors. A rather simplistic statistical model is often invoked; see Henrici [4], page 305, ff. Even more elaborate discussions can be found, as in Rademacher [7]. For the simplistic model, the argument is roughly as follows. If an error of  $\delta$  could occur at each operation, and if the error should occur in the same direction each time, then in  $n$  operations a total error of  $n\delta$  would accumulate. However, if the errors should fluctuate randomly between positive and negative, then in  $n$  operations a total error of  $k\sqrt{n}\delta$  would (usually) occur. People even try to determine a suitable value to assign to  $k$ . However, this is hardly justified.

Consider, if we have  $n$  positive summands all about the same size, and all too small by  $\delta$ , the error after  $n$  additions should be  $n\delta$ ; the sum will also be about  $n$  times each individual term, so that the percent of error has not increased (the distributive law again). Actually, the way most calculators work, the error after  $n$  additions could be much more than  $n\delta$  for very large  $n$ . Review again the example with the divergent series. If  $n$  is large enough, one can reach the point where adding another number makes no change. Then the error increases much faster than by  $\delta$  at each step. However, if the addition is properly

arranged (see discussion below), the distributive law will be reasonably well approximated, and the error will indeed be about  $n\delta$ , to be compared with a sum about  $n$  times the individual summands.

Suppose that, because of randomness, the error is claimed to be  $k\sqrt{n} \delta$ . Suppose that a value has been assigned to  $k$ , so that with the given value of  $n$  the error is claimed to be only  $n\delta/10$ . If the summands are all about the same size, the sum is about  $n$  times an individual term. So we claim an accuracy for the sum ten times as good as for the individual terms. If each summand was entered to maximum accuracy on an 8-place calculator, then it would have the sum correct to 9 places, which is obviously impossible.

However, there are many cases where the general distribution of errors is random enough that the predictions of the statistical theory seem reasonably well fulfilled. The total error (in some sense) grows at a rate proportional to  $\sqrt{n}$  rather than to  $n$ . For additional exceptions, see the next section.

In Crary and Rosser [5], power series coefficients for 41 functions were calculated to high accuracy; between 40 and 50 coefficients were computed for each function. In the report, these coefficients were uniformly rounded to considerably fewer decimals at print out than were carried in the calculation. Among the checks of accuracy of transcription, the sum of the coefficients for each function was also computed to high accuracy.

For each function, the transcribed and rounded coefficients were added to see how their sum compared with the true sum (see page 48 of Cray and Rosser [ 5]). The results are shown in Table 1. This is a reasonable approximation to a Gaussian distribution.

Sum too high by	Number of cases
+4	2
+2	3
+1	9
0	12
-1	9
-2	2
-3	4

Table 1.

We might remark that in summing (3.1) we used the worst possible procedure. It is standard doctrine that, in summing a series, rounding error will accumulate more slowly if one adds starting with the small numbers. The reason for this is that if one adds the large terms first, then when a small term is added in, one must truncate it or round it to bring it to the right number of decimals, and information is lost. If one adds several small terms together first, this information is used (in part) to form an accurate partial sum of a size more comparable to the large terms. Still

better, if the series is suitably graduated, would be to keep the partial sums more nearly of a size with each other. In the series (3.1), this could be done very nicely by adding the terms in pairs, then adding the pairs in pairs, etc. With this system, one could possibly get a sum of any part of the series (3.1) to about the full accuracy which the machine could accomodate.

If one wishes to calculate the determinant of a large matrix, one can be subject to both cancellation and rounding errors. By definition, the determinant involves a large number of differences, so that there is great opportunity for cancellation errors. It involves a large number of terms, so that rounding errors can accumulate badly. Closely related is the problem of solving a large number of simultaneous linear equations. With proper techniques, one can sidestep some of the difficulties. However, it remains a problem in which even the best methods can occasionally fail disastrously. A very full discussion is given in Kahan [8].

#### 4. A particular example. Define

$$(4.1) \quad \varepsilon = 0.000013516 \text{ (octal).}$$

Then

$$(4.2) \quad \varepsilon \cong 0.0000 \ 44450 \ 16384 \text{ (decimal) .}$$

Let us enter  $\epsilon$  into a calculating machine with 9-place octal accuracy (floating), compute  $r = 1 - \epsilon$ , and then erase the value of  $\epsilon$ . How well can the computer recover  $\epsilon$  from  $r$ ?

As an 8-place decimal machine is nearly equivalent to a 9-place octal machine, we will write much of our discussion as though for an 8-place decimal machine; this machine is assumed to be designed in such a manner as to truncate prior to addition, so that machine error also occurs.

The first thing is to make sure we subtract  $\epsilon$  properly from 1. The machine will hold 1 as

$$1.0000000,$$

being an 8-place machine. To subtract  $\epsilon$ , it will truncate it to

$$0.0000444.$$

Subtracting will give  $r \cong 0.99995560$ . Already, we have lost valuable information.

What we do is to calculate

$$r = \left(\frac{1}{2} - \epsilon\right) + \frac{1}{2}.$$

This will give

$$(4.3) \quad r \cong 0.99995555.$$

From this, how good a value of  $\epsilon$  can we compute? If we carelessly take

$$\epsilon = 1 - r,$$

we will get the approximation  $0.445 \times 10^{-4}$ . If we remember to take

$$\epsilon = (0.99999 - r) + (0.00001)$$

we can get the approximation  $0.4445 \times 10^{-4}$ . This ought to be the best we can do. However, note that

$$(4.4) \quad 1 - r = \frac{1 - r^N}{1 + r + r^2 + \dots + r^{N-1}}.$$

Take  $N$  large, say  $N = 262,144$ . Then, with the given value of  $r$ , we have  $r^N < 10^{-5}$ . So, if we can give an approximation for  $r^N$  good to 3 significant decimal digits, we will have the numerator good to 8 significant digits; that is, if we remember to take

$$1 - r^N = \left(\frac{1}{2} - r^N\right) + \frac{1}{2}.$$

There is no cancellation in the denominator of (4.4). So we should be able to get it accurate to about 8 significant digits. So then (4.4) should give  $\epsilon$  good to nearly 8 significant digits!

Of course, one has to be smart about summing the denominator, as discussed near the end of the previous section. We carried out a number of calculations on the UNIVAC 1108, which has 9-place octal

accuracy (floating). In each case, just to compare results, we summed the denominator of (4.4) in three different ways: summing from the left, summing from the right, and iterated summing by pairs.

The first question is about the calculation of  $r^N$ . The analysis given in Wilkinson [3] applies perfectly to this calculation. If the rounding is biased, the accumulation of rounding errors could be so severe that we would get an approximation for  $r^N$  correct to only two significant decimal digits. In fact, after multiplying, the UNIVAC 1108 truncates instead of rounding, thus introducing machine error. After 262,144 multiplications by  $r$ , we got  $r^N \approx 8.684 \times 10^{-6}$ , as contrasted with the more accurate value of

$$(4.5) \quad r^N \approx 8.696\ 388\ 5 \times 10^{-6}$$

(got by a double precision calculation). However, even this poor value would give  $1-r^N$  with an error of only 2 units in the 8-th decimal digit.

However, one cannot expect the denominator of (4.4) to be very good, with truncation instead of rounding at each step. We got the results shown in Table 2. Considering the bias in the rounding, these are not too bad.

Methods of summing the denominator	$\varepsilon \times 10^5$
true value	4.445 0164
summing from the left	4.450 5777
summing from the right	4.446 0810
iterated summing by pairs	4.445 5325

Table 2.



In Yohe [ 9 ], it is shown how various sorts of rounding, including various types of "ideal" rounding, could be accomplished by simple hardware or simulated by software on most computers. We programmed the UNIVAC for ordinary rounding, thus eliminating machine error as defined by (1.2). Presumably the statistical theory should have some validity in this case. Indeed, we did far better. We got  $r^N \approx 8.696\ 3897 \times 10^{-6}$ . So we have full accuracy for the numerator of (4.4). We got the results shown in Table 3. It appears that we have recovered 7 of the 8 digits of  $\epsilon$ , and are off only 3 units in the 8-th place.

Methods of summing the denominator	$\epsilon \times 10^5$
true value	4.445 0164
summing from the left	4.445 3544
summing from the right	4.445 0177
iterated summing by pairs	4.445 0167

Table 3.

As long as we are looking for convenient forms for the calculation of  $\epsilon$ , we might note that since  $N = 2^{18}$ , we can write

$$(4.6) \quad 1 - r = \frac{1 - r^{2^{18}}}{(1+r)(1+r^2)(1+r^4)(1+r^8)\dots(1+r^{2^{17}})}$$

This involves very few multiplications and additions, and should be still better. However, it is not. It gives 4.445 1465 as an approximation for  $\varepsilon \times 10^5$ .

The reason for this appears if we give a little thought to the matter. We computed  $r^2, r^4, r^8, \dots$  by successive squarings. If  $A = (1+\varepsilon)r^M$  is an approximation for  $r^M$ , then

$$A^2 = (1 + 2\varepsilon + \varepsilon^2)r^{2M}.$$

Of course, in general,  $A^2$  will not be a number that can be stored in the machine. The result of multiplying  $A$  by  $A$  on the machine will be  $B$ , where

$$B = (1 + \delta)A^2.$$

So the approximation  $B$  for  $r^{2M}$  is related to  $r^{2M}$  by

$$B = (1+\delta)(1+2\varepsilon+\varepsilon^2)r^{2M}.$$

If  $\delta$  and  $\varepsilon$  have opposite signs,  $B$  could be as good an approximation for  $r^{2M}$  as  $A$  was for  $r^M$ , possibly even better. However, with reasonable randomness of rounding errors, there will come a time when  $\delta$  and  $\varepsilon$  have the same sign. So we will from time to time get a fairly large  $\varepsilon$ , and eventually the  $2\varepsilon$  term will become overriding. From then on, we essentially double the percent

error at each squaring. This is an example of numerical instability, which we mentioned earlier as a possible source of error. The value of  $r^N$  as calculated by successive squaring on the UNIVAC 1108 with rounding was approximately  $8.693\ 4161 \times 10^{-6}$ . This is not a whole lot better than the value derived by 262,144 successive multiplications by  $r$  with truncation at each step.

5. Rounding error depends on the number base used in the calculator. But of course! However, the effect can be much greater than one might expect. For example, the black magic in the previous section, in which we were apparently recovering almost eight digits of  $\varepsilon$  from  $1-\varepsilon \cong 0.99995555$ , was wholly an illusion caused by the differences in rounding between a 9-digit octal machine and an 8-digit decimal machine. Recall that we took

$$\varepsilon = 0.0000\ 13516 \text{ (octal)}$$

exactly. Then

$$r = 0.7777\ 64262 \text{ (octal)}$$

exactly. So

$$(5.1) \quad 1-r = 0.0000\ 13516\ 0000 \text{ (octal)}$$

exactly. The formula (4.4), when applied properly, will give this value to nearly 9 significant octal digit accuracy. Converted to decimal, this means we were getting close to

$$0.0000\ 4445\ 0164$$

for  $\epsilon$ ; the illusion was created that we were recovering nearly 8 decimal digits of  $\epsilon$ .

As another illustration of a startling divergence due to differences of rounding, recall that when we calculated  $r^N$  by successive squaring, using octal rounding, we got

$$8.693\ 4161 \times 10^{-6}$$

as compared with a more accurate value of

$$r^N \cong 8.696\ 3885 \times 10^{-6}.$$

However, if we start with the decimal equivalent of  $r$ ,

$$r \cong 0.99995555,$$

and perform successive squarings with decimal rounding, we get

$$r^N \cong 8.696\ 0578.$$

If used in (4.6), this would have given

$$\epsilon \cong 4.445\ 0305 \times 10^{-5}.$$

If this paper has a moral, it is that cancellation and rounding errors are more serious than is generally believed, especially if one runs into some numerical instability. Mitigation of these effects due to randomness of errors is not as trustworthy as one might believe from the statistical theories that have been propounded. However, the remedy used by many people, of going to a double precision calculation if any suspicion of unreliability appears, will probably be entirely adequate in all but an extremely small minority of cases. Nonetheless, even with double precision, it is worthwhile giving thought to whether, of several mathematically equivalent forms, one has chosen the one least likely to produce cancellation errors. Also, if one has a sum involving a large number of summands, it is worthwhile, and not much trouble, to give thought to the best order of performing the summation.

#### REFERENCES

- [1] J. Barkley Rosser, "Potentials of charged plates, Part I,"  
MRC-TSR #1318, April 1973.
- [2] J. Barkley Rosser and N. Papamichael, "A power series  
solution of a harmonic mixed boundary value problem,"  
MRC-TSR #1405, January 1975 and Technical Report TR/35,  
Brunel University, December 1973.

- [ 3] J. H. Wilkinson, "Rounding errors in algebraic processes,"  
National Physical Laboratory Notes on Applied Science, no. 32,  
London, HMSO, 1963.
- [ 4] Peter Henrici, "Elements of numerical analysis," John Wiley  
and Sons, Inc., New York, 1964.
- [ 5] Fred D. Crary and J. Barkley Rosser, "High precision coefficients  
related to the zeta function," Mathematics Research Center  
Technical Summary Report #1344, May 1, 1973.
- [ 6] W. Kahan, "A survey of error analysis," pp. 1214-1239,  
Information Processing 71, Proceedings of International  
Federation of Information Processing Congress 1971, Ed. C. V.  
Freiman, ed., vol. II, North Holland Publishing Company, 1972.
- [ 7] Hans A. Rademacher, "On the accumulation of errors in  
processes of integration on high-speed calculating machines,"  
pp. 176-187, Annals of the Computation Laboratory of Harvard  
University, vol. XVI, Harvard University Press, 1948.
- [ 8] W. Kahan, "Numerical linear algebra," Canadian Mathematical  
Bulletin, vol. 9 (1966), pp. 757-801.
- [ 9] J. Michael Yohe, "Roundings in floating-point arithmetic,"  
IEEE Transactions on Computers, vol. C-22 (1973), pp. 577-586.
- [10] J. Michael Yohe, "Implementing a nonstandard data type,"  
MRC-TSR #1545, June, 1975.

APPLICATIONS OF NUMERICAL MODELING  
TO COASTAL ENGINEERING PROBLEMS

H. Lee Butler and D. L. Durham  
U. S. Army Engineer Waterways Experiment Station  
P. O. Box 631, Vicksburg, Miss. 39180

**ABSTRACT.** A review of the numerical modeling efforts within the Wave Dynamics Division, Hydraulic Laboratory at the Waterways Experiment Station is presented. Numerical modeling has progressed rapidly in the last several years and is now generally recognized as a useful tool capable of yielding valuable information on alternative solutions to many coastal engineering problems. Present efforts include models for application to: Wind-driven circulation; storm surge on a lake; tsunami generation and transoceanic propagation; tidal hydraulics of bays, harbors, and inlets; generation and propagation of landslide generated water waves; and harbor oscillations, both free and forced. A discussion of the mathematical formulations and applications is given herein. Future efforts are discussed which include stratified lake circulation and both deep and shallow water wind wave generation.

**1. INTRODUCTION.** The advent of large scale computer systems has made it possible to use hydrodynamic theory based upon rational physical approximations rather than the dictates of mathematical tractability. Constraints still exist due to computer limitations and lack of basic understanding of certain phenomena; however, numerical modeling has progressed rapidly in the last several years. The mathematical model is now generally recognized as a tool capable of yielding valuable information on the particular phenomenon under investigation.

All of the models discussed in this paper are based on long wave theory approximations. There are many problems to which long wave theory can be applied and the degree of validity of each application is dependent on the particular approximation made and must be carefully evaluated. Long wave theory is a valid approximation when the ratio of water depth to wave length is small and the vertical component of motion does not significantly influence the pressure distribution which is assumed to be hydrostatic.

A wide range of applications together with model formulations are presented. It is not the intent of this paper to present detailed descriptions of models, but, instead, to indicate the level of effort being undertaken by various members of the staff at the Waterways Experiment Station. The reader is asked to consult the appropriate reference to obtain details of the model formulations.

---

This paper was presented at the 1975 Army Numerical Analysis and Computers Conference.

2. SCOPE OF MODELING EFFORTS. Dynamic prediction equations based on the fundamental differential equations of hydrodynamics can be applied with minor variations to provide data on wave motion where the depth of water is small relative to the wave length. A linearized version of the model expressed in a spherical coordinate system has been used to calculate the propagation of tsunamis (originating from seismic disturbances in 12 segments of the Aleutian Trench) from their source to the west coast of the United States. The results were used to evaluate the relative vulnerability of an area along the west coast to tsunami inundation as a function of seismic location.

A model based on the same fundamental equations, but cast into a Cartesian coordinate system, has been applied in analyzing the tidal hydraulics associated with harbors and inlets. Additional terms including non-linear inertial forces (convective forces) and bottom stress were considered. The numerical tidal model has been used as a complementary tool in the operation of a physical model of the Los Angeles and Long Beach harbor complex. Such a numerical model can provide valuable information for use in the design and operation of physical models. Some of the areas to which the numerical model can be of assistance are documented below:

- a. Establishing appropriate boundaries during design of the physical model which are sufficiently removed from the influence of regions to be investigated.
- b. Indicating regions where data collection may be more critical or desirable.
- c. Indicating possible problem areas in advance of physical model tests.
- d. Providing assistance in the general interpretation of physical model results for certain phenomena.
- e. Providing a "quick look" at a number of possible configurations indicating the more promising plans.
- f. Providing a mechanism for investigating certain phenomena such as wind which cannot be easily introduced into the physical model.

The integration of the numerical and physical wave models allows for a more complete and comprehensive test program.

Under a project entitled, "General Investigation of Tidal Inlets" an attempt has been made to construct an idealized inlet in order to investigate the effects of inlet geometry, friction, and bay and tidal heights. An effort was undertaken to investigate the ability of the numerical model to reproduce localized velocity patterns. The results bore out the fact that many details of the velocity patterns seen in the physical model were reproduced in the numerical model. Applications to actual inlets have been made and comparisons with prototype elevation and velocity data obtained.



With minor variations the tidal model has been applied in the investigation of landslide generated water waves. In 1971, the Hydraulics Laboratory at WES constructed a 1:120 undistorted scale model of Libby Dam and Lake Koocanusa, Montana, and conducted tests to determine wave heights, runup and amount of overtopping resulting from the sliding of individual rock ribs into the reservoir. Subsequently a study was initiated at WES and directed toward developing numerical methods for predicting the effects of landslide generated water waves in reservoirs. The agreement between the physical and numerical models was very good considering that each set of results (five separate slides with different slide velocities) were subject to certain inherent limitations imposed by the model theory and others imposed by lack of knowledge concerning prototype conditions or how to model certain phenomena.

The Lake Erie Regional Transportation Authority is conducting a feasibility and site selection study for a major hub airport in the Cleveland service area. One of the possible sites being evaluated is an offshore site near Cleveland, Ohio. As a part of the feasibility analysis of an offshore site, WES is conducting a model feasibility investigation and a part of this study is to determine the effects of the jetport on the hydrodynamics of the surrounding area. Two of the numerical models applied so far in the project have been used to define the effects of the jetport on the following phenomena:

- a. Near-field and far-field definition of the wind-driven circulation for constant density, well mixed lake conditions.
- b. Storm surge at the shoreline from Lorain to Fairport, Ohio.

For many coastal engineering problems, particularly in the design of harbors, the capability of estimating the response of partially enclosed bodies of water such as harbors, bays, etc., to long wave excitation can be a useful tool for preliminary investigations as well as a guide to physical model studies. The geometric shape of the harbor and the wave reflections associated with its boundaries produce amplification or attenuation of the incidence wave regime. This phenomenon is often referred to as harbor resonance or seiching and can excite adverse motion of moored ships and produce currents which may be hazardous to navigation. New harbors or modifications to existing harbors can be designed to minimize the effects of harbor resonance by mismatching the frequencies at which a harbor has maximum response and the frequencies of the pre-dominant long wave energy which is characteristic of the wave regime at the harbor site.

The major problem is how to determine accurately the amplitude distribution function and frequencies of the resonance characteristics of a harbor. One approach is the application of numerical techniques. Two methods have been applied at WES:

- a. A finite element method for obtaining the two-dimensional amplitude distribution and frequencies of undamped, natural modes of oscillations.
- b. The solution of a boundary value problem for long wave-induced oscillations (forced oscillations).

These methods were applied to the harbor at Port Hueneme, California, in conjunction with a physical long wave model study of the harbor. An additional application of the free oscillation model was made to Lake Erie in order to determine the effects of a jetport structure on the free oscillations of Lake Erie.

3. TSUNAMI MODEL. A computational scheme relating arbitrary sea-floor displacements and the consequent surface-wave history was developed by Hwang, Butler, and Divoky (1972). In order to permit more reliable estimates of far-field surface displacements as well as ease of including the effects of the earth's curvature, the problem was cast into a spherical grid covering the area of interest.

The governing equations can be written as:

#### Momentum Equations

$$(1) \quad \frac{\partial U}{\partial t} = - \frac{g}{r_e} \frac{\partial \eta}{\partial \theta}$$

$$(2) \quad \frac{\partial V}{\partial t} = - \frac{g}{r_e \sin \theta} \frac{\partial \eta}{\partial \phi}$$

#### Continuity Equation

$$(3) \quad \frac{\partial \eta}{\partial t} = - \frac{1}{r_e \sin \theta} \left\{ \frac{\partial}{\partial \theta} ((h+\eta)U \sin \theta) + \frac{\partial}{\partial \phi} ((h+\eta)V) \right\} + \frac{\partial \varepsilon}{\partial t}$$

where the spherical coordinate system is described in Figure 1a.

To solve these equations an alternating direction technique is employed similar to that of Leendertse (1967). A space-staggered scheme is used in which velocities, water level, and depth are described at different points within a grid cell (Fig. 1b). The first half cycle of the calculation consists of computing  $U$  and  $\eta$  implicitly and  $V$  explicitly, advancing from time  $n\Delta t$  to  $(n + \frac{1}{2}) \Delta t$ . The second half cycle computes  $\eta$  and  $V$  implicitly and  $U$  explicitly, advancing from time  $(n + \frac{1}{2}) \Delta t$  to  $(n + 1) \Delta t$ .

By continuing these processes one can determine the values of  $U$ ,  $V$ , and  $\eta$  as functions of time. The finite difference equations involved in the first half cycle of the calculation are:

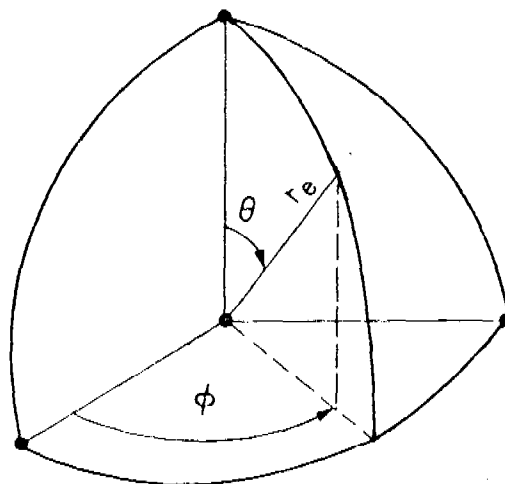
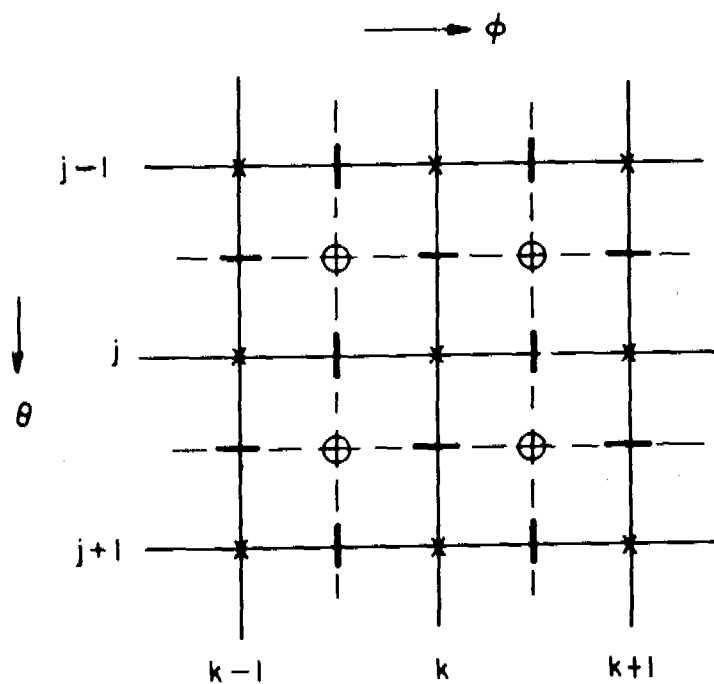


Fig. 1a. Spherical coordinate system



- WATER DEPTH  $h$
- x WATER LEVEL  $\eta$
- $u$  VELOCITY IN THE DIRECTION  $\theta$
- |  $v$  VELOCITY IN THE DIRECTION  $\phi$

Fig. 1b. Space-staggered grid definition

$$(4) \quad U_{j+\frac{1}{2},k}^{n+\frac{1}{2}} = U_{j+\frac{1}{2},k}^n - \frac{\Delta t g}{2r_e \Delta \theta} (\eta_{j+1,k}^{n+\frac{1}{2}} - \eta_{j,k}^{n+\frac{1}{2}})$$

$$(5) \quad \eta_{j,k}^{n+\frac{1}{2}} = \eta_{j,k}^n - \frac{\Delta t}{2r_e \sin \theta_j} \left\{ \frac{1}{\Delta \theta} \left( \{(\overline{h+\eta})U \sin \theta\}_{j+\frac{1}{2},k}^{n+\frac{1}{2}} - \{(\overline{h+\eta})U \sin \theta\}_{j-\frac{1}{2},k}^{n+\frac{1}{2}} \right) \right.$$

$$\left. + \frac{1}{\Delta \phi} \left( \{(\overline{h+\eta})V\}_{j,k+\frac{1}{2}}^n - \{(\overline{h+\eta})V\}_{j,-\frac{1}{2}}^n \right) \right\} + \Delta t / 2 \left( \frac{\partial \epsilon}{\partial t} \right)^n$$

These equations can be solved for  $U^{n+\frac{1}{2}}$  and  $\eta^{n+\frac{1}{2}}$  where terms noted with a bar are computed by averaging values of  $h$  and  $\eta$  to determine the barred quantities at the specific points. The explicit expression for  $V^{n+\frac{1}{2}}$  is

$$(6) \quad V_{j,k+\frac{1}{2}}^{n+\frac{1}{2}} = V_{j,k+\frac{1}{2}}^n - \frac{\Delta t g}{2r_e \Delta \phi \sin \theta_j} (\eta_{j,k+1}^n - \eta_{j,k}^n)$$

The difference equations for the second half cycle are similar with  $U$  and  $V$  exchanging roles.

Two types of boundary conditions are involved in the computations. These are the solid boundaries at coastlines and the open sea boundaries arising from the need to truncate the region of computation to minimize computer core and processing time requirements. A condition of complete reflection is adopted at solid boundaries. No dissipative factor is considered in the model at present to account for loss of tsunami energy due to shoreline interaction. This condition is accomplished by setting  $U=0$  or  $V=0$  at the appropriate boundary.

The open sea condition must simulate a total transmission of the wave through the boundary. This cannot be rigorously achieved without computation beyond the boundary and thus an artifice is adopted which assumes that the wave profile travels without change of form across the last interior cell of the grid at the shallow-water wave speed. This leads to the expression

$$(7) \quad \eta_B^{n+\frac{1}{2}} = (\eta_{B-1}^n - \eta_B^n) \frac{\Delta t \sqrt{gh}}{\Delta S} + \eta_B^n$$

where  $B$  refers to an exterior face of a boundary cell,  $B-1$ , the corresponding interior face of the same cell, and  $\Delta S$  the dimension of the cell normal to the boundary face.

The derivative of the ground motion term appearing in the continuity equation accounts for the time dependent vertical bottom displacement measured from a pre-quake bottom topography. If far-field results are the major interest, an initial water surface deformation is assumed instead

of allowing the numerical code to simulate the time dependent vertical motion. This initial surface condition conforms to the topographic features of the ocean floor which experiences permanent vertical displacement resulting from a quake disturbance. Since the characteristic time for vertical ground movement is small when compared to the long periods of tsunami waves, it is sufficient to assume an instantaneous uplift over the entire disturbance area.

The tsunami model was verified by hindcasting the 1964 Alaskan Tsunami (Hwang, et al; 1972). With observations of generated bottom displacements and distant wave histories available, the computed and observed waves compared well in amplitude and phase of the dominant leading wave.

Results of the Aleutian Trench study (Whalin, Garcia, and Butler; 1974) indicate that the tsunami generated by the 1964 Alaskan quake may not be a rare event at all. Further, a tsunami of equal intensity resulting from a seismic disturbance located elsewhere in the Aleutian Trench could cause significantly higher runup at particular west coast locations. Figure 2 displays a sample plot of surface elevation contours at a time of 4 hours following a hypothetical disturbance in segment 7 of the Aleutian Trench (a source location south of Shumagin Island).

4. TIDAL MODEL. A numerical tidal model, based upon the original formulation by Leendertse (1967), has been applied at WES to analyze the tidal hydraulics of harbors and inlets. Again the fundamental equations are averaged over the vertical dimension and expressed in a Cartesian coordinate system to yield

#### Momentum Equations

$$(8) \quad \frac{\partial U}{\partial t} + U \frac{\partial U}{\partial x} + V \frac{\partial U}{\partial y} + g \frac{\partial \eta}{\partial x} - fV + \frac{1}{h+\eta} (T_B - T_w)^x = 0$$

$$(9) \quad \frac{\partial V}{\partial t} + U \frac{\partial V}{\partial x} + V \frac{\partial V}{\partial y} + g \frac{\partial \eta}{\partial y} + fU + \frac{1}{h+\eta} (T_B - T_w)^y = 0$$

#### Continuity Equation

$$(10) \quad \frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x} \{ (h+\eta)U \} + \frac{\partial}{\partial y} \{ (h+\eta)V \} = 0$$

where the stress terms are defined as

#### Bottom stress

$$(11) \quad T_B^x = \frac{gU}{C^2} \sqrt{U^2 + V^2}$$

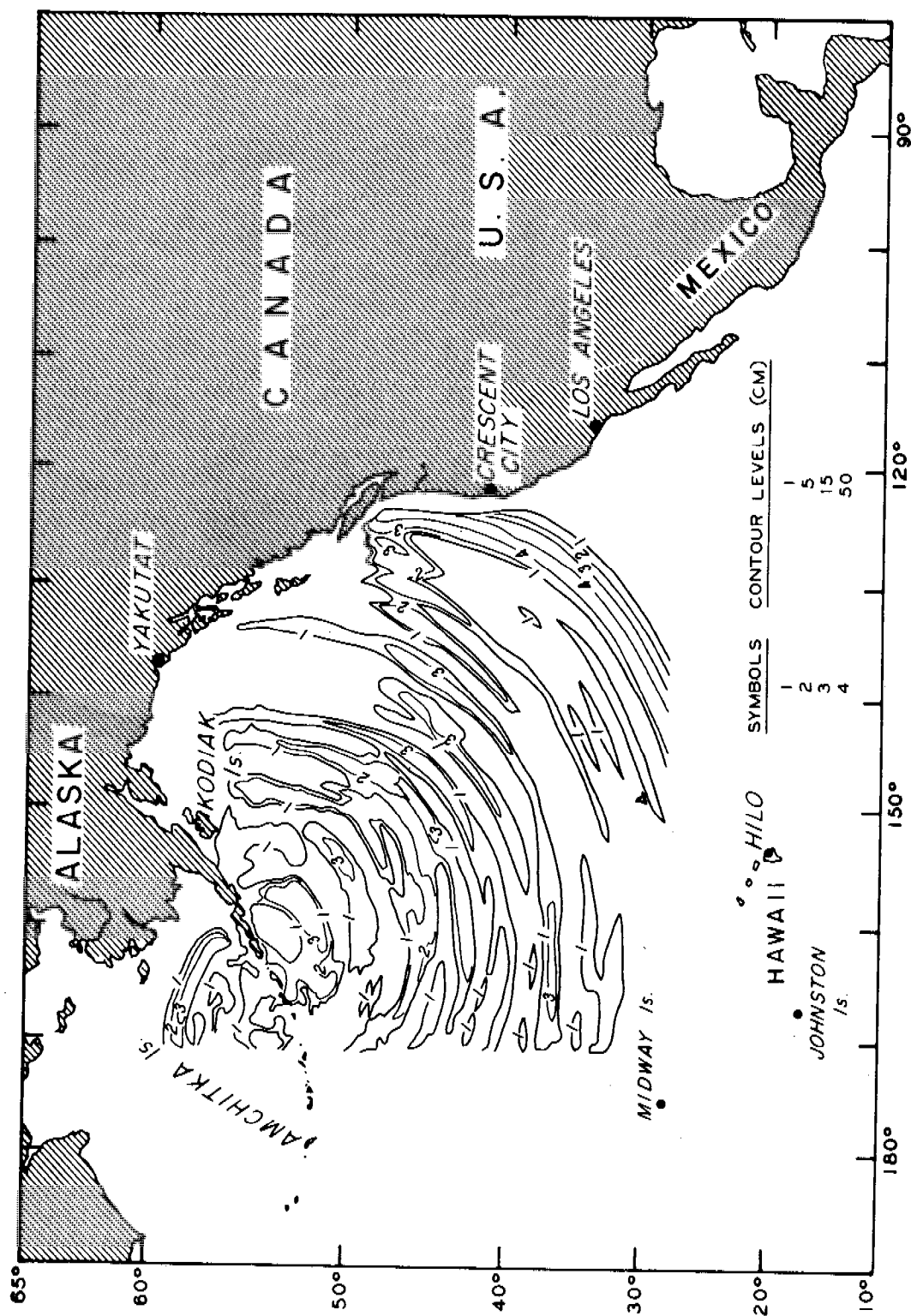


Fig. 2. Surface elevation contours resulting from a hypothetical disturbance

and

$$(12) \quad \tau_B^y = \frac{gV}{C^2} \sqrt{U^2 + V^2}$$

Wind stress

$$(13) \quad \vec{\tau}_w = C_d \rho_a \vec{W}_a^2$$

The major assumptions involved along with the consequence of each assumption are outlined below:

- a. Internally the fluid is inviscid (horizontal and vertical eddy viscosity terms are considered to be negligible).
- b. The fluid is incompressible (continuity equation and gravity term are simplified).
- c. The vertical component of the fluid acceleration is negligible (momentum equation in the vertical direction reduces to the hydrostatic pressure variation).
- d. The fluid is well mixed so that the variations of flow in the vertical direction are small (allowing the averaging of flow quantities over the vertical).

To solve the above system of equations numerically the differential equations must be discretized and replaced by a system of finite difference equations using central differences on a space-staggered grid as in the case with the tsunami model. Again, Leendertse's implicit-explicit multi-operational method is employed in determining the solution for  $U$ ,  $V$ , and  $\eta$  as functions of time.

The difference equations for the second half cycle are written as:

$$(14) \quad \begin{aligned} V^{n+1} = & V^{n+\frac{1}{2}} - \frac{g\Delta t}{2\Delta y} \langle \eta_y \rangle^{n+1} - \frac{\Delta t}{4\Delta x} \bar{U}^{n+\frac{1}{2}} \langle V_x \rangle^{n+\frac{1}{2}} \\ & - \frac{\Delta t}{4\Delta y} V^{n+1} \langle V_y \rangle^{n+\frac{1}{2}} - \frac{\Delta t}{2} F_y \quad \text{at } j, k+\frac{1}{2} \end{aligned}$$

$$(15) \quad \eta^{n+1} = \eta^{n+\frac{1}{2}} - \frac{\Delta t}{2\Delta x} \langle (h+\eta)U \rangle_x^{n+\frac{1}{2}} - \frac{\Delta t}{2\Delta y} \langle (h+\eta)V \rangle_y^{n+1} \quad \text{at } j, k$$

These equations are solved for  $V^{n+1}$  and  $\eta^{n+1}$  along a grid line  $j$ . For simplicity of notation some terms have been maintained in differential form within angle brackets  $\langle \rangle$ . Central differences are used in evaluating these terms. The additional velocity component  $U^{n+1}$  can be determined explicitly from the equation

$$(16) \quad \begin{aligned} U^{n+1} = & U^{n+\frac{1}{2}} - \frac{g\Delta t}{2\Delta x} \langle \eta_x \rangle^{n+\frac{1}{2}} - \frac{\Delta t}{4\Delta x} U^{n+1} \langle U_x \rangle^{n+\frac{1}{2}} \\ & - \frac{\Delta t}{4\Delta y} V^{n+1} \langle U_y \rangle^{n+\frac{1}{2}} - \frac{\Delta t}{2} F_x \quad \text{at } j+\frac{1}{2}, k \end{aligned}$$

The expressions for the forcing terms  $F_x$  and  $F_y$  are written as

$$(17) \quad F_x = f\bar{V}^{n+1} + \frac{1}{(h+n)^{n+1}} \{ (\tau_x^w)^{n+1/2} - \frac{gU^{n+1}}{\bar{c}^2} \{ (U^{n+1/2})^2 + (\bar{V}^{n+1})^2 \} \}$$

$$(18) \quad F_y = -f\bar{U}^{n+1/2} + \frac{1}{(h+n)^{n+1/2}} \{ (\tau_y^w)^{n+1/2} - \frac{gV^{n+1/2}}{\bar{c}^2} \{ (\bar{U}^{n+1/2})^2 + (V^{n+1/2})^2 \} \}$$

In addition the terms signified by a single bar are not defined at the point where the equation is written and are determined by averaging two neighboring values. The terms signified by a double bar require four neighboring values to obtain an average.

Three types of boundaries are involved in the calculations:

- a. A condition of complete reflection is considered at solid boundaries.
- b. Tidal elevations are input data at artificial ocean boundaries.
- c. Tidal elevations or average fluid velocities are input data at artificial boundaries in the inlet, harbor or bay.

Sufficient prototype data must be available to operate and/or verify the model. Along these lines, it should be emphasized that tidal elevations or average fluid velocities must be defined at all artificial boundaries in the inlet, harbor or bay.

From a consideration of the assumptions involved in the development of the model, it is apparent that it should be most applicable for considering long period waves in non-stratified regions where vertical accelerations are small. The model provides no indication of the vertical distribution of velocity. Phenomena which depend on the detailed velocity distribution as a function of the depth cannot be investigated.

As mentioned previously, applications of the model at WES include the Los Angeles-Long Beach Harbor area and an idealized inlet model. Figure 3 illustrates the tidal velocity field for a portion of the LA-LB harbor. The treatment of breakwaters and piers on piles are of particular interest in this application. The frictional coefficients were varied to simulate the resistance of these structures to the flow. Approximately 1600 grid points were used in the computational grid.

5. LANDSLIDE MODEL. A numerical model for the generation and propagation of landslide generated water waves in reservoirs was developed at WES (Raney and Butler; 1975) as part of an effort directed toward



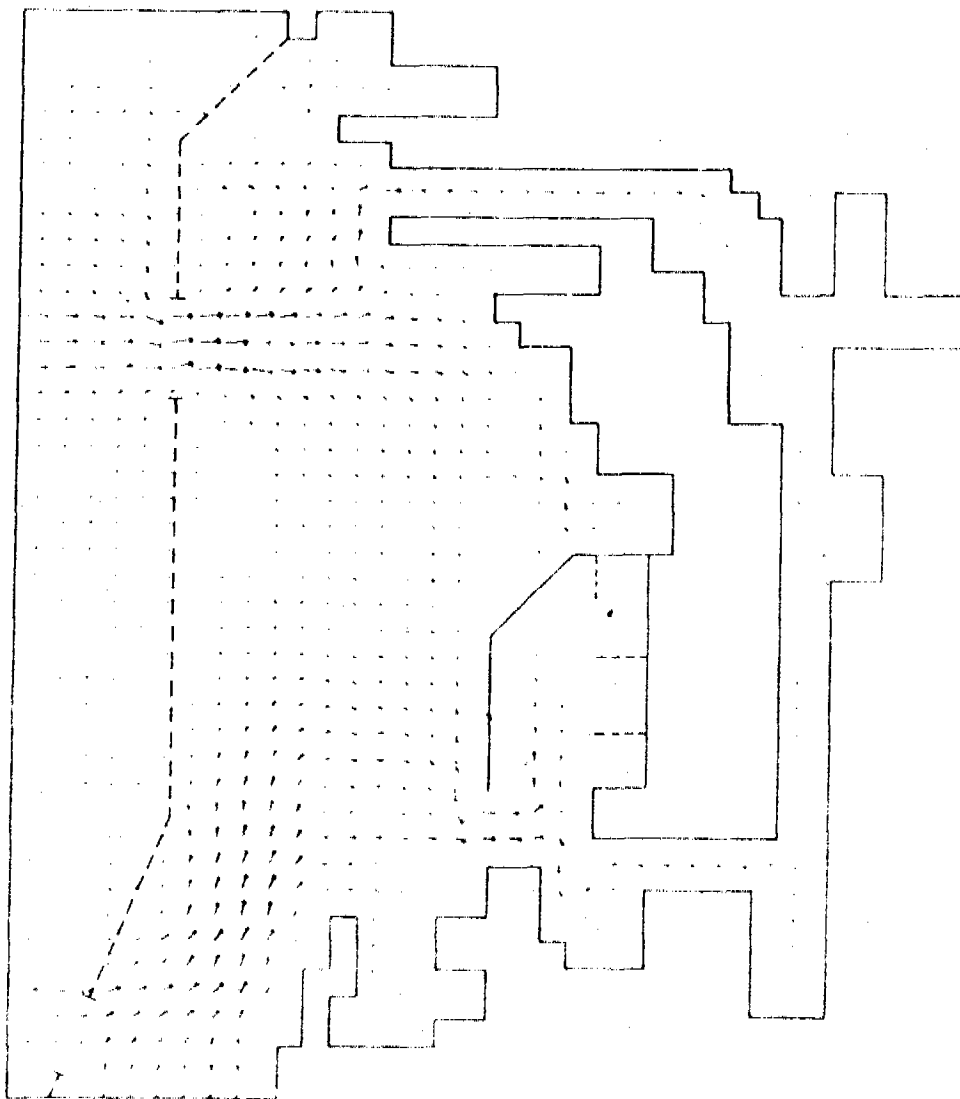


Fig. 3. Illustration of a tidal velocity flow field for a portion of the Los Angeles-Long Beach Harbor Complex

finding analytical and/or numerical methods for predicting the effects of these waves. It was decided that a promising initial approach for the investigation involved using a depth averaged formulation of the fluid mechanics equations.

The tidal model as discussed previously can be adopted with appropriate modifications to evaluate the arrival time and height of the first wave crest which may result from a landslide. Results from the physical model study conducted at WES (Davidson and Whalin; 1974) indicated that in most cases the first wave crest was the largest measured at each of the model gauges. The pertinent modifications to the tidal model are:

- a. A vertical bottom deformation term representing the passage and settling of landslide material is introduced into the continuity equation in a manner similar to the ground motion term appearing in the tsunami model.
- b. The acceleration effect of the landslide on the fluid with which it is in contact is introduced into the momentum equations as part of the forcing terms.

Thus the equations can be written as:

#### Momentum Equations

$$(19) \quad \frac{\partial U}{\partial t} + U \frac{\partial U}{\partial x} + V \frac{\partial U}{\partial y} + g \frac{\partial \eta}{\partial x} = R_x + L_x$$

$$(20) \quad \frac{\partial V}{\partial t} + U \frac{\partial V}{\partial x} + V \frac{\partial V}{\partial y} + g \frac{\partial \eta}{\partial y} = R_y + L_y$$

#### Continuity Equation

$$(21) \quad \frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x} \{ (h+\eta)U \} + \frac{\partial}{\partial y} \{ (h+\eta)V \} = \frac{\partial \epsilon}{\partial t}$$

The method of solution is identical to that used in the tidal model.

Boundary conditions are similar to those adopted in the tsunami model. The fictitious open boundaries arise from the need to truncate the region of computation (in order to minimize computational time requirements). Also considered is a time dependent boundary between the landslide surface and the water in the reservoir.

The landslide is represented by a time dependent deformation of the bottom of the reservoir plus additional terms to represent the effect of the landslide on the reservoir due to viscous and inertia forces. The bottom deformation propagates into specified regions of the reservoir at

the average speed of the landslide with the deformation at any particular location increasing from zero to a maximum value according to a specified time-displacement relationship. For those portions of the bottom of the reservoir through which the landslide passes but which do not experience a net change in ground elevation as a result of the final slide deposition, the deformation is allowed to return to zero at a specified rate. The handling of the landslide condition is illustrated in Fig. 4. The direction, extent, and magnitude of the bottom deformation is determined by knowledge or assumptions concerning the path and final disposition of the particular landslide.

The water in the reservoir experiences an acceleration due to the force exerted by the landslide at the time dependent boundary between the water and the landslide. This force per unit mass is considered to consist of a component due to the displacement of the water by the slide plus two components which act on fluid elements in contact with the landslide:

$$(22) \quad L_x = \alpha(v_x - U)^2 + \beta (v_x - U)^2$$

$$(23) \quad L_y = \alpha(v_y - V)^2 + \beta (v_y - V)^2$$

where

$v_x$  = the x component of the landslide velocity

$v_y$  = the y component of the landslide velocity

$\alpha$  = viscous drag parameter

$\beta$  = pressure drag parameter

The first component of  $L_x$  and  $L_y$  is related to the viscous drag exerted by the slide upon the fluid with which it is in contact. The second component of  $L_x$  and  $L_y$  expresses a pressure drag exerted on the water by the front of the slide. The viscous drag is considered to act at all points of contact between the slide and the water. The pressure drag acts only at the leading face of the landslide.

The landslide is thus represented numerically by a combination of terms whose net effect in the numerical model should be a reasonable representation of the large scale physical effects produced by a landslide entering a reservoir. A representation of the landslide propagates into the numerical representation of the reservoir, accelerating the fluid due to physical displacement, viscous effects, and pressure drag effects. The resulting waves propagate across the reservoir in accordance with the governing equations. The wave height and velocity components are calculated for each grid cell at the end of each time step.

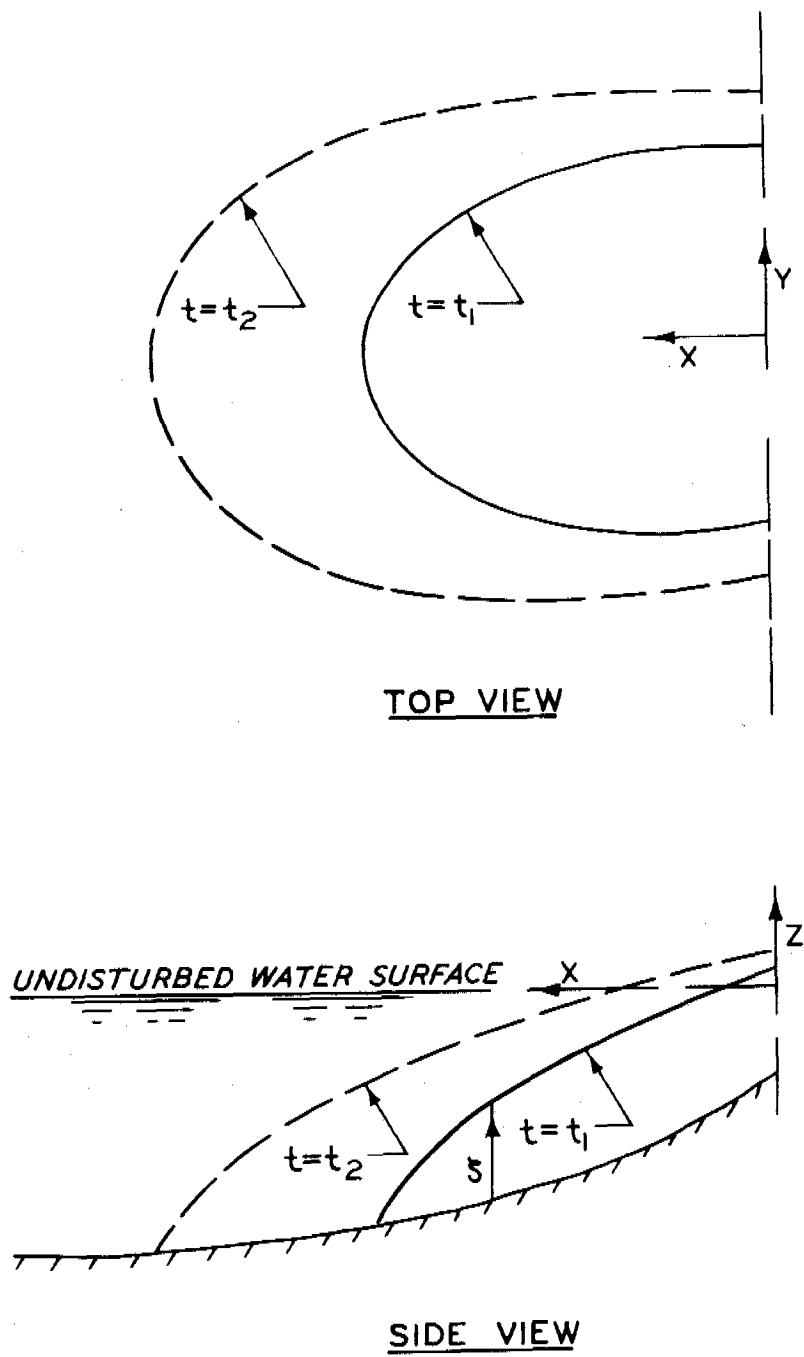


Fig. 4. Landslide model

The numerical model was verified by comparing computed results with the experimental data from the physical model of Libby Dam and Lake Kooncanusa. Figure 5 shows the physical model ready for a test run. The landslide material used in the model tests was 1/18 cu ft bags filled with lead and iron ore. Figure 6 shows the final position of a slide.

Defining the landslide characteristics is the most critical aspect of the model. It is necessary to know or assume the magnitude of the slide material, the average velocity at which it moves, its path through the water, the general shape of its leading face, a time-vertical displacement relationship for the slide and the final disposition of the slide in the reservoir. In a general investigation the use of this model would require a parametric study. Available for this study were experimental data obtained in a physical model study so that necessary parameters were known or could be approximated to a reasonable degree of accuracy. This study then reflects the degree to which the mathematical model can represent the reservoir conditions if the slide characteristics are reasonably well defined.

Good agreement existed between the numerical model and the physical model results. No real attempt, once the numerical formulation of the model was completed, was made to manipulate the input parameters to improve the agreement with the experimental data from the physical model.

6. STEADY STATE WIND-DRIVEN CIRCULATION MODEL. A relatively realistic and accurate model of the steady state, constant density, wind-driven currents in Lake Erie has been developed by Gedney and Lick (1972). The analysis is based on Welander's shallow lake model (Welander, 1957), which is an extension of Ekman's oceanic model. The basic assumptions in the analysis are:

- a. steady state,
- b. constant density,
- c. vertical variation of pressure is hydrostatic,
- d. horizontal diffusion is neglected,
- e. non-linear convection is neglected, and
- f. constant coefficient of vertical diffusion.

These assumptions allow the three-dimensional equations of motion to be integrated analytically in the vertical direction and, in combination with the continuity equation, give a two-dimensional elliptic equation for the integrated stream function.

Having made the assumptions listed above, the basic equations can be written as:

Fig. 5. Physical model of Libby Dam and Lake Koocanusa  
ready for a test run



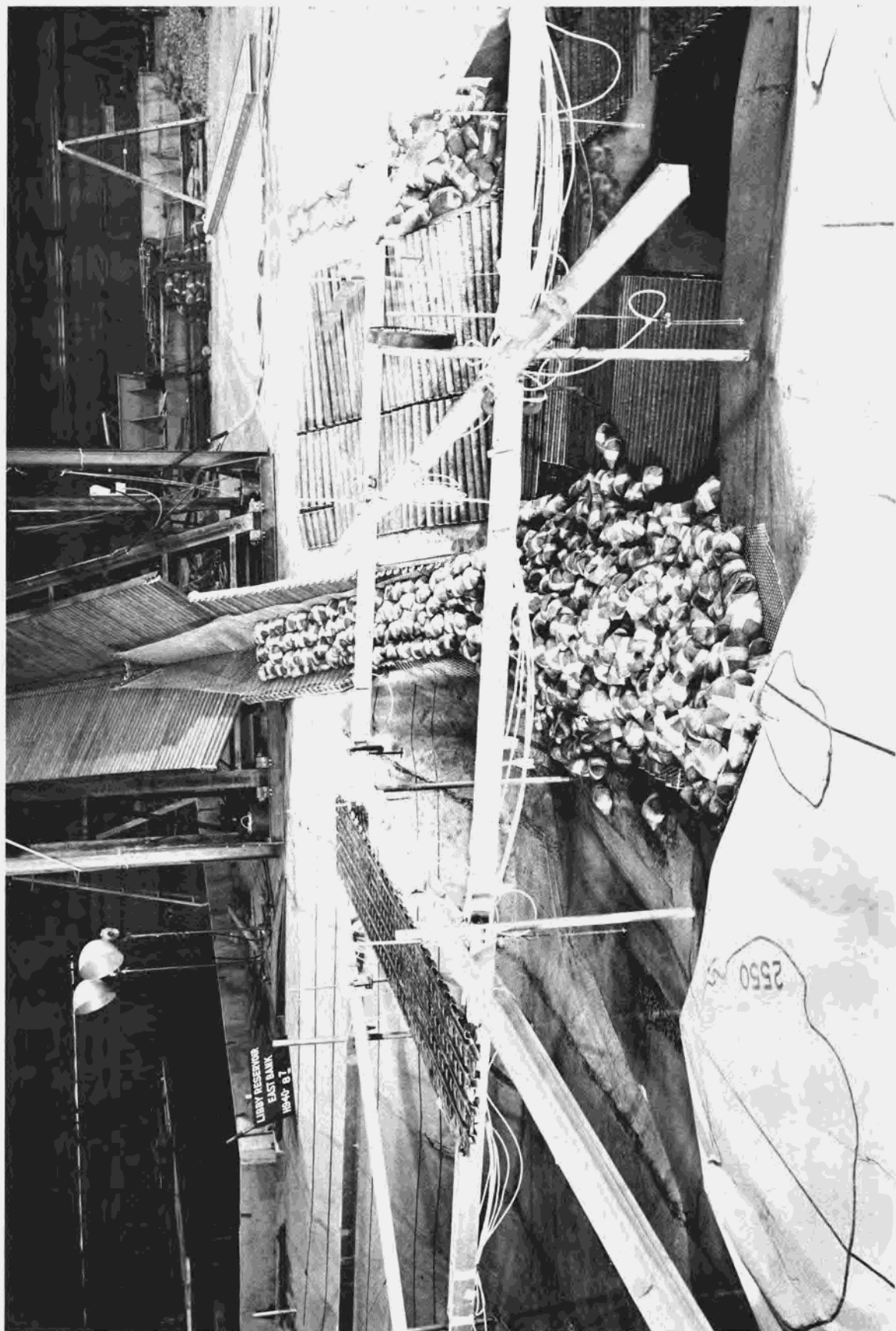


Fig. 6. Final position of slide material after a test run

### Equations of Motion

$$(24) \quad E_v \frac{\partial^2 u}{\partial z^2} + v = \frac{\partial \eta}{\partial x}$$

$$(25) \quad E_v \frac{\partial^2 v}{\partial z^2} - u = \frac{\partial \eta}{\partial y}$$

### Continuity Equation

$$(26) \quad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

All the variables involved have been non-dimensionalized by appropriate quantities. The associated boundary conditions are:

$$(27) \quad u = v = w = 0 \quad \text{at the lake bottom}$$

$$(28) \quad T_x^w = K \frac{\partial u}{\partial z} \quad \text{at the still lake level}$$

$$(29) \quad T_y^w = K \frac{\partial v}{\partial z}$$

The equations of motion can yield analytic expressions for  $u$  and  $v$  as functions of  $\frac{\partial \eta}{\partial x}$ ,  $\frac{\partial \eta}{\partial y}$ ,  $T_x^w$ ,  $T_y^w$ ,  $h$ , and  $z$ . The expressions for  $u$  and  $v$  are complex and are given by Gedney and Lick (1972). By integrating these expressions for  $u$  and  $v$ , together with the continuity equation, over the lake depth, expressions for  $\frac{\partial \eta}{\partial x}$  and  $\frac{\partial \eta}{\partial y}$  can be determined as functions of the mass transport and wind stress. By defining

$$(30) \quad U = \frac{\partial \psi}{\partial y} \text{ and } V = -\frac{\partial \psi}{\partial x}$$

where  $\psi$  is the integrated stream function, a second order differential equation for  $\psi$  can be derived and expressed as

$$(31) \quad \nabla^2 \psi + \gamma_1 \frac{\partial \psi}{\partial x} + \gamma_2 \frac{\partial \psi}{\partial y} = \gamma_3$$

where the coefficients  $\gamma_1$ ,  $\gamma_2$ , and  $\gamma_3$  are functions of depth, bottom slope, and wind stress. The stream function is specified on the boundary.

A successive over-relaxation (SOR) technique is used to solve the associated second order elliptic difference equation. Once the stream function is known, the previous expressions for  $u$  and  $v$  can be evaluated, and integrating the continuity equation will yield the vertical velocity component  $w$ .



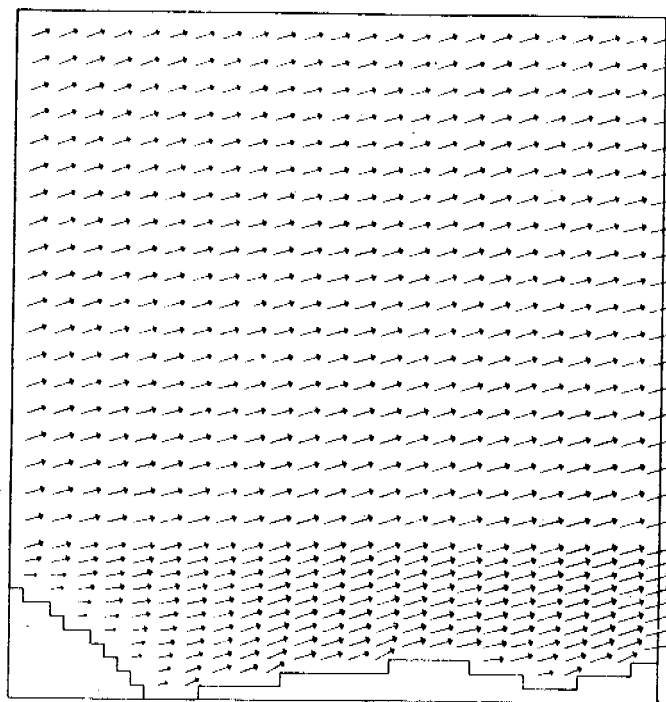


Fig. 9. Without a jetport structure

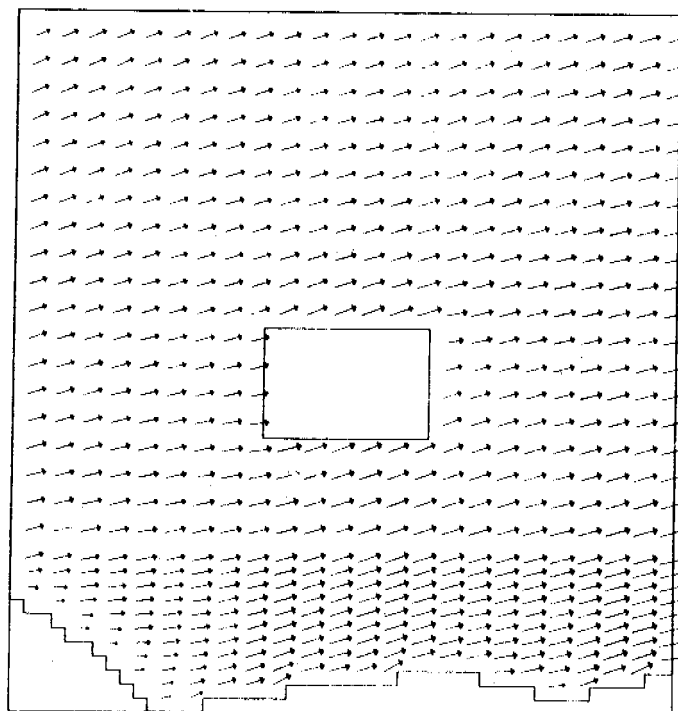


Fig. 10. With a jetport structure, 5 miles offshore  
Steady-state horizontal velocity field in the Cleveland area  
at the surface

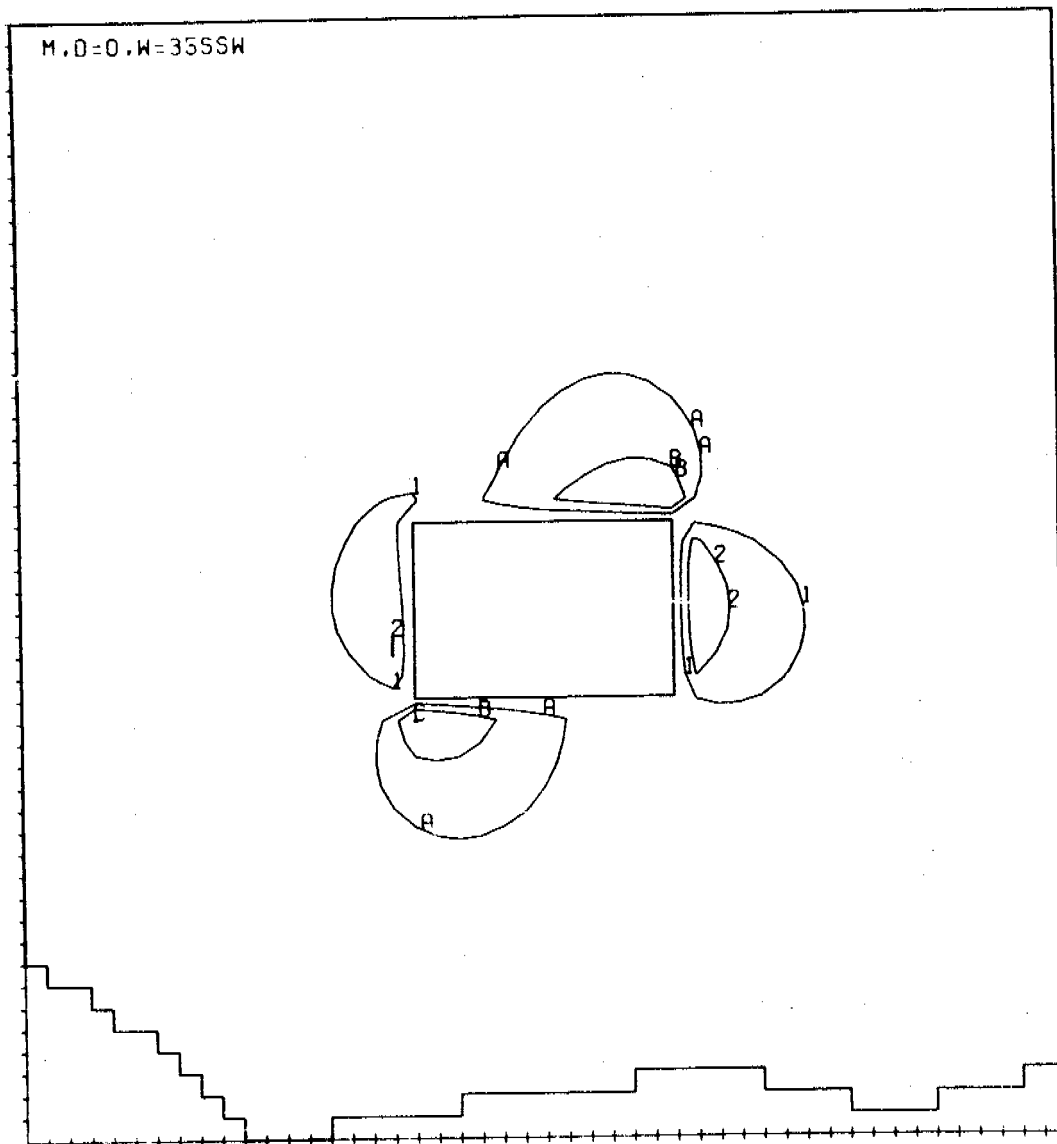


Fig. 11. Contour plot of differences in velocity magnitudes  
with and without the jetport structure

- a. A finite element method (FEM) used to find the natural oscillation solution for an open or closed basin, and
- b. a method to estimate the two-dimensional amplitude distribution function of wave-induced (forced) oscillations in harbors of complex geometry as well as the wave frequencies at which maximum harbor response occurs.

In theory, both types of oscillations can be mathematically represented by boundary value problems formulated from the long wave equations and appropriate boundary conditions.

### The Finite Element Method

Assuming a periodic form of the solution of the linear long wave equation for a variable depth basin, the governing differential equation for undamped harmonic oscillations of a variable depth basin becomes

$$(32) \quad \frac{\partial}{\partial x} \left\{ h(x,y) \frac{\partial \eta(x,y)}{\partial x} \right\} + \frac{\partial}{\partial y} \left\{ h(x,y) \frac{\partial \eta(x,y)}{\partial y} \right\} + \lambda^2 \eta(x,y) = 0$$

where

$$(33) \quad \lambda^2 = \omega^2 / g$$

The boundary condition for this problem is that the normal component of velocity at the basin boundary be equal to zero. Since equation (32) (special form of the Helmholtz equation) is the equation for a standing wave, this boundary condition can be expressed as

$$(34) \quad \frac{\partial \eta(x,y)}{\partial n} = 0$$

where

$n$  = unit normal to boundary

For the open basin case the boundary along the basin opening cannot be satisfied by the above boundary condition since this is not the physical case. The surface elevation along such an opening is assumed to be a nodal line. Thus, the boundary condition along the open boundary is

$$(35) \quad \eta(x,y) = 0$$

Therefore, equations (32-35) are the partial differential equation and the boundary conditions for the boundary value problem of the natural oscillations of a body of water. It can be shown that the  $x$  and  $y$  components of velocity ( $u,v$ ) are proportional to the  $x$  and  $y$  components of the horizontal gradient of surface elevation.

Solutions to this boundary value problem for basins of complex geometry and variable topography have been estimated numerically by Abel (1970)\*. He used a calculus of variations approach and obtained an Euler-Lagrange formulation of the boundary value problem. The variational formulation for the boundary value problem basically consists of the functional integral

$$(36) \quad \chi = \iint_A \frac{1}{2} \{ h(x,y) \left[ \frac{\partial \eta(x,y)}{\partial x} \right]^2 + h(x,y) \left[ \frac{\partial \eta(x,y)}{\partial y} \right]^2 - \lambda^2 \eta^2(x,y) \} dx dy$$

which, for a stationary value of  $\chi$  (maximum or minimum) with respect to  $\eta(x,y)$ , the Euler-Lagrange condition of the functional is identically equal to equation (32). In addition to this functional integral, the Euler-Lagrange formulation possesses appropriate boundary conditions. The solution  $\eta(x,y)$ , which maximizes or minimizes equation (36), is obtained by utilizing finite-element techniques. Details of these techniques can be found in most finite-element texts (i.e., Desai and Abel, 1972) and will not be pursued here. Concisely, the finite-element technique, as utilized in this problem, is a numerical approximation procedure for evaluating the integral equation obtained from maximizing the functional integral. The surface area of the study region,  $A$ , is divided into  $N$  subregions,  $A^e$  (elements) with  $e = 1, \dots, n$ , with the shape of the element (triangle or quadrangle) being defined by a number of nodes (vertices) which connect the element to other elements of the grid. The integral equation is expressed and applied at the element level and later assembled for the total domain of interest. This procedure produces a system of  $M$  (total number of node points) linear algebraic equations in terms of the  $M$  values of  $\eta_m$  at the nodal points and in which the values of  $\eta_m$  form the solution of equation (32). This system of equations can be expressed in the form of an eigenvalue problem and solved by matrix procedures for the eigenvalues (natural frequencies of the system) and the eigenvectors (configuration of relative magnitudes of surface displacements at each nodal point).

#### Forced Oscillation Method

The boundary value problem for long wave-induced oscillations in arbitrary shaped harbors of constant depth,  $h$ , is governed by the following differential equation

$$(37) \quad \frac{\partial^2 \eta(x,y)}{\partial x^2} + \frac{\partial^2 \eta(x,y)}{\partial y^2} + k^2 \eta(x,y) = 0$$

which is called the Helmholtz equation, where the wave number  $k$  is defined as

$$(38) \quad k = \omega / \sqrt{gh}$$

\* WES internal Memorandum for Record.

Lee (1969) in mathematically solving this boundary value problem divided the area of interest into two regions, a harbor region and an infinite ocean region, having the same constant depth,  $h$ . For each region, the Helmholtz equation is the governing differential equation; therefore, the problem in each region is to determine the wave function  $\eta(x,y)$  which satisfies this differential equation and the appropriate boundary conditions. In the ocean region, the solution of the wave function is assumed to be composed of three separate parts:

- a. An incident wave,  $\eta_i(x,y)$
- b. A reflected wave,  $\eta_r(x,y)$ , from the coastline with harbor entrance closed and
- c. A radiated wave,  $\eta_3(x,y)$ , emanating from the harbor entrance.

Physically, the wave regime in the ocean region is composed of a steady-state standing wave regime with a radiated wave due to the presence of the harbor superimposed on it.

The boundary conditions for both regions are:

- a. Along all solid boundaries such as the harbor boundary and the coastline, the normal velocity component is zero or

$$(39) \quad \frac{\partial \eta(x,y)}{\partial n} = 0$$

- b. Along the harbor entrance, a "continuity" or matching condition is used such that the wave amplitude and slope of the water surface obtained from the solution in the ocean region must equal these quantities obtained from the solution in the harbor region.
- c. At the harbor entrance an outwardly radiated wave is generated by the harbor and diminishes in magnitude as the seaward distance from the harbor's entrance increases such that at an infinite distance from the harbor there is no effect of the harbor on the wave regime in the ocean.

The latter condition allows some energy in the harbor to be dissipated by radiating energy into the ocean, thus limiting the amplification of the harbor response at resonance. This condition also governs the selection of the mathematical form of the fundamental solution of the Helmholtz equation in the ocean region and likewise in the harbor region. The form of the fundamental solution chosen is the Hankel function,  $H_0^1(kr)$  where

$$(40) \quad r = \sqrt{x^2 + y^2}$$

This form is chosen because it satisfies the Helmholtz equation, decays to zero at infinity and possesses the proper type of singularity at the origin.

Using Green's identity formula and the fundamental solution  $[H_0^{(1)}(kr)]$  of the two-dimensional Helmholtz equation, a solution for the wave function,  $\eta(x,y)$ , at any position in the domain of interest is obtained in integral form as a function of the value of  $\eta(x,y)$  and  $\frac{\partial \eta(x,y)}{\partial n}$  at the boundary.

$$(41) \quad \eta(\vec{x}) = -\frac{i}{4} \int \left\{ \eta(\vec{x}_0) \frac{\partial}{\partial n} [H_0^{(1)}(kr)] - H_0^{(1)}(kr) \frac{\partial}{\partial n} [\eta(\vec{x}_0)] \right\} ds(\vec{x}_0)$$

This form of solution for the Helmholtz equation is known as the Weber solution, and its derivation is presented in potential theory texts such as Baker and Copson (1950). To determine the wave function at any interior point in the region of interest, the value of  $\eta(x,y)$  and the value of  $\frac{\partial \eta(x,y)}{\partial n}$  on the boundary of the region are required. Lee and Raichlen (1972) showed that a matrix approximation of an integral equation similar in form to equation (41) could be used to obtain  $\eta(x,y)$  along the boundary in terms of  $\frac{\partial \eta(x,y)}{\partial n}$  along the boundary. With the boundary condition that the normal derivative of the wave function on the solid boundary is zero, the wave function on the boundary and, in turn, the wave function at any interior point can be obtained as a function of the unknown normal derivative of the wave function along the entrance of the harbor. In a similar manner as the harbor region, the wave function at any interior point of the ocean region is obtained as a function of the unknown normal derivative of the wave function along the harbor entrance. Therefore, by matching the solution in each region at the harbor entrance, the unknown normal derivatives of the wave function at the harbor entrance can be obtained. Thus, the wave function on the boundary can be calculated and the wave function at any position in the harbor can be determined. Then, the amplification factor for the response of the harbor can be calculated by taking the ratio of the wave amplitude at any position in the harbor to the incident plus reflected wave amplitude at the coastline with the entrance closed.

#### Applications of Harbor Oscillation Models

During a physical long wave model study at WES of Port Hueneme in California, a review and evaluation of existing theories of wave-induced oscillations of harbors were conducted with some limited application to Port Hueneme of Lee and Raichlen's numerical method as well as Abel's numerical procedure for natural modes of oscillation. This study allowed WES personnel an opportunity to become familiar with the limitations and applicability of these numerical models and to establish their use as a

guide to the operation of physical long wave models of harbors. For Port Hueneme, the frequencies and modal shapes of the natural oscillations of the harbor were obtained from Abel's method (Fig. 12). Then, the wave-induced harbor response was calculated by Lee and Raichlen's procedure (Fig. 13) for the frequency range of harbor resonance as established by Abel's finite-element method. The results of both approaches were compared with results of the physical model (Fig. 14). The horizontal configuration of the harbor response compared very well between the three procedures with the exception of the lower reaches of the entrance channel near the harbor entrance. The variations in the results of each method can reasonably be explained by the limitations and restrictions of each procedure. Based on this study, the numerical procedures can provide estimates of the frequencies at which harbor resonance may occur. This information allows the physical model to be operated at a lower resolution of wave frequencies in the frequency ranges where no resonant peak is expected and to increase the frequency resolution near the resonant frequencies. This procedure allows more efficient operation of the physical model. The location of instruments in the physical model in areas of maximum surface elevation changes and areas of maximum velocities can be facilitated by the results of these numerical procedures. For some of the resonant frequencies, this may eliminate a set of wave tests in the model to define the relative configuration of the harbor response. The limitations and assumptions of these numerical/analytical methods must be kept firmly in mind when using them as a guide to the physical wave modeling procedure.

An additional application of the FEM was made in the Lake Erie Jetport study. The finite-element grid contained 237 elements and 262 nodes with representative depth values at each node as shown in Fig. 15a. A comparison of the first five computed natural frequencies with observed values by Platzman showed very good agreement (Fig. 15b shows the fundamental mode). Also, comparisons of computed relative amplitudes to observed values at 13 locations around the lake were very good (Table 1).

**9. FUTURE EFFORTS IN NUMERICAL MODELING.** Future numerical modeling efforts at WES will include the applications of models not discussed in this paper as well as further attempts to extend the applications discussed herein. A synopsis of these efforts is given below:

- a. Seismic disturbances in the Peru-Chile Trench area off the Chilean coast are considered capable of producing tsunamis resulting in significant runup in southern California. A comprehensive set of calculations will be made for this area as a follow up to the Aleutian Trench study.
- b. The tidal model discussed herein does not address itself to the flooding of tidal flats, marsh areas, etc. In addition to considering inundation, features such as a representation of weir sections and overtopping barriers will also be

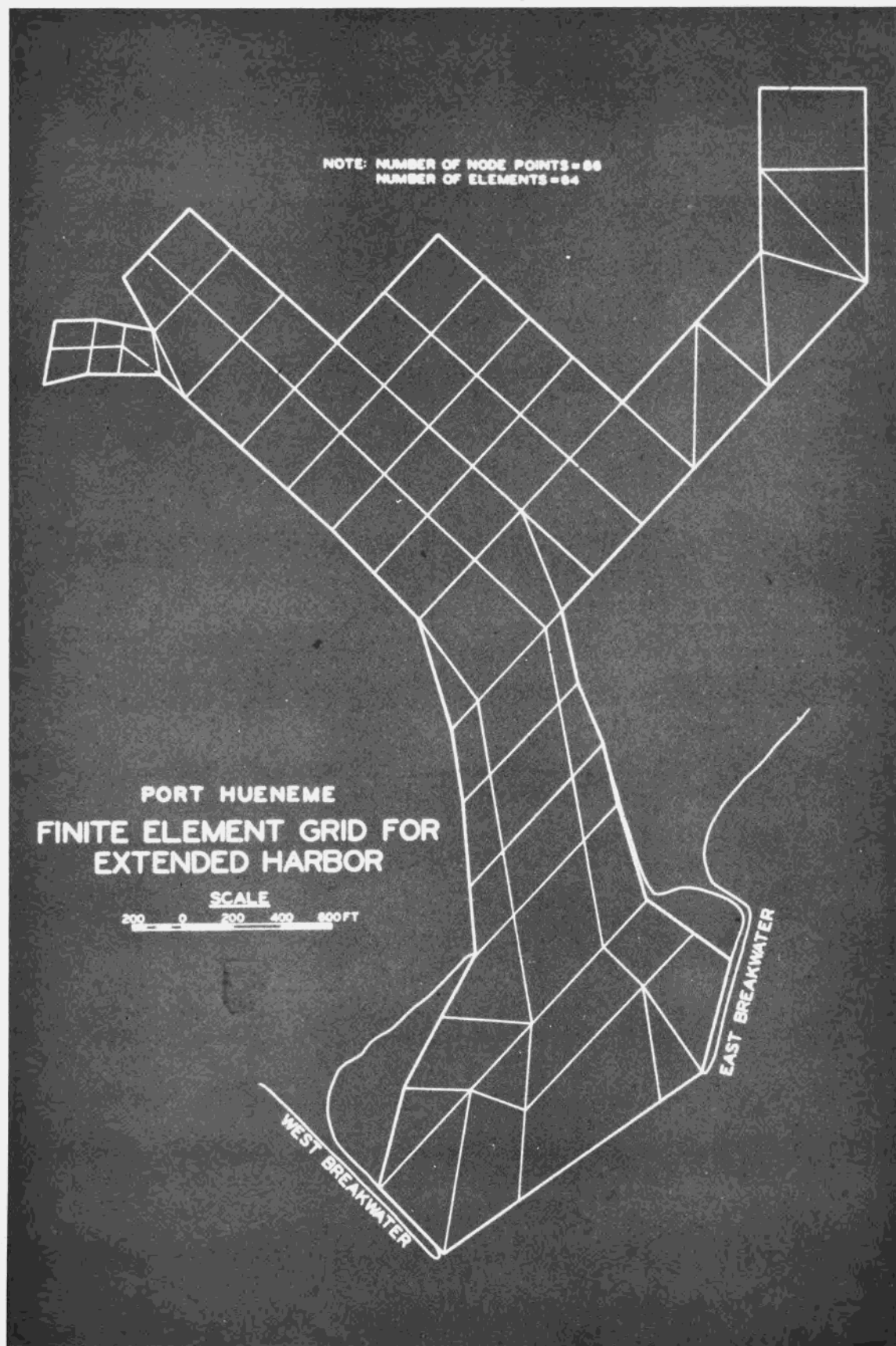


Fig. 12. Finite element grid for Port Hueneme



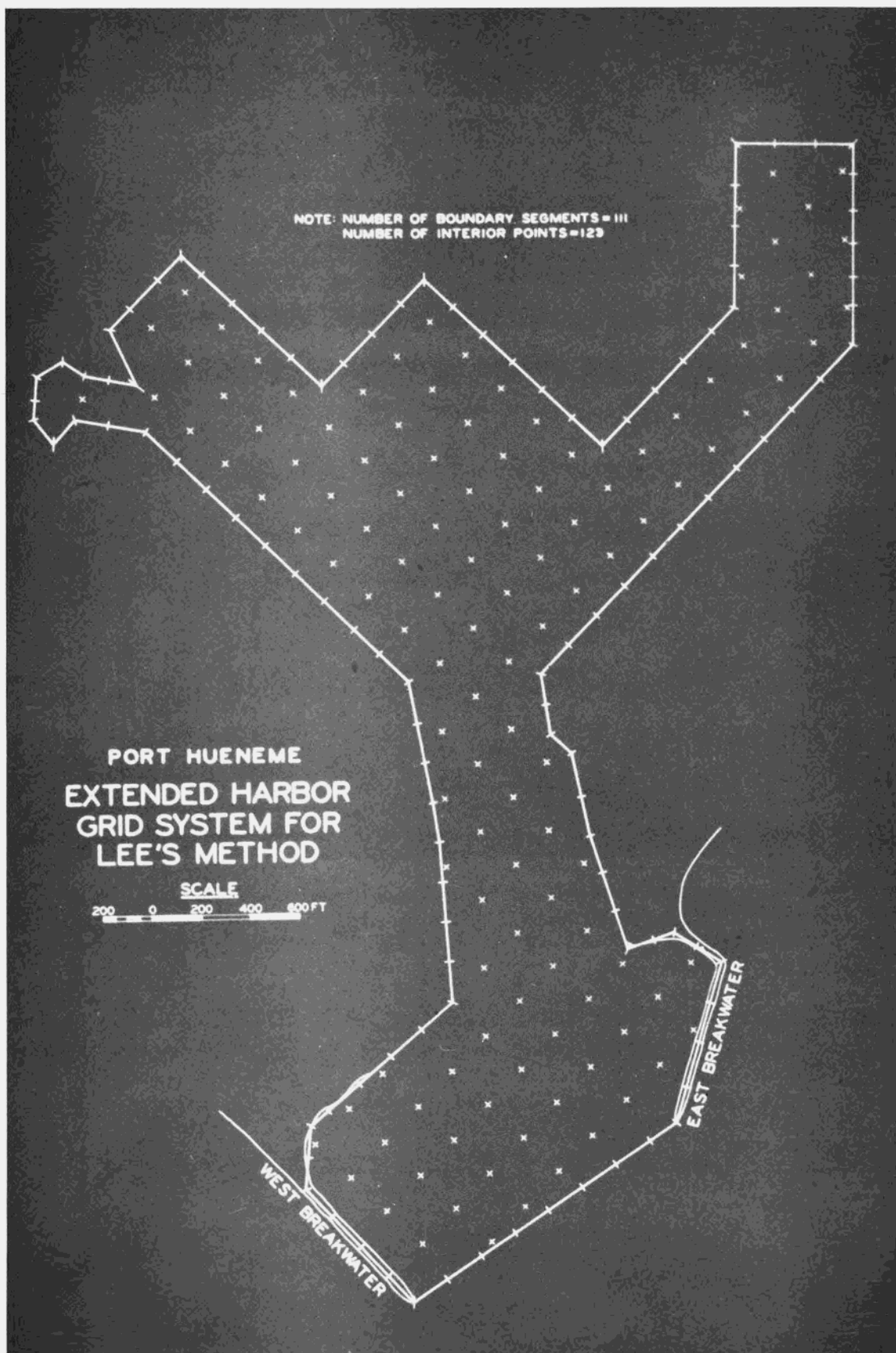


Fig. 13. Grid used in Lee-Raichlen procedure

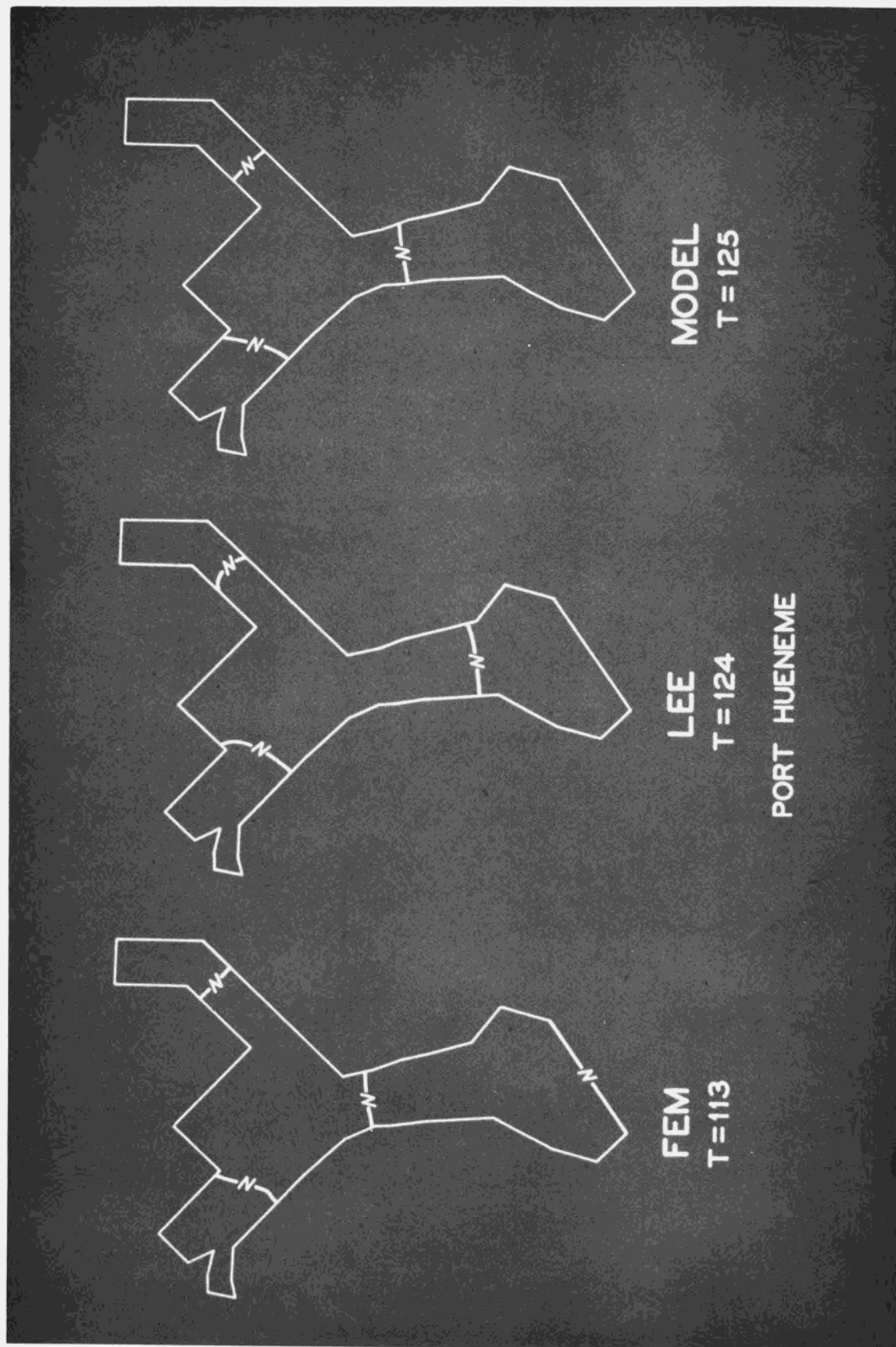


Fig. 14. Comparison of calculated and measured nodal lines for the fundamental mode of oscillation

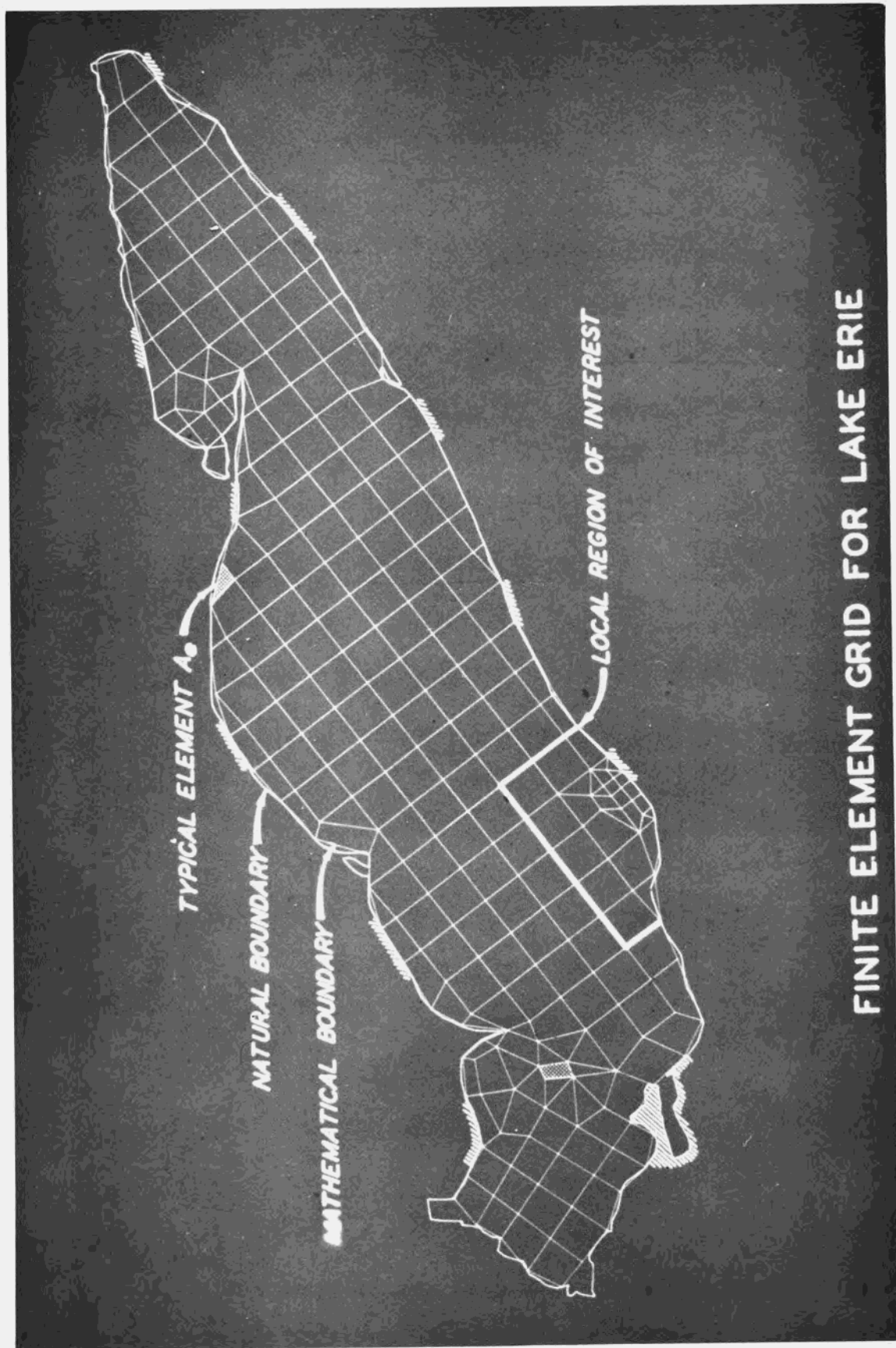


Fig. 15a. Finite element grid for Lake Erie

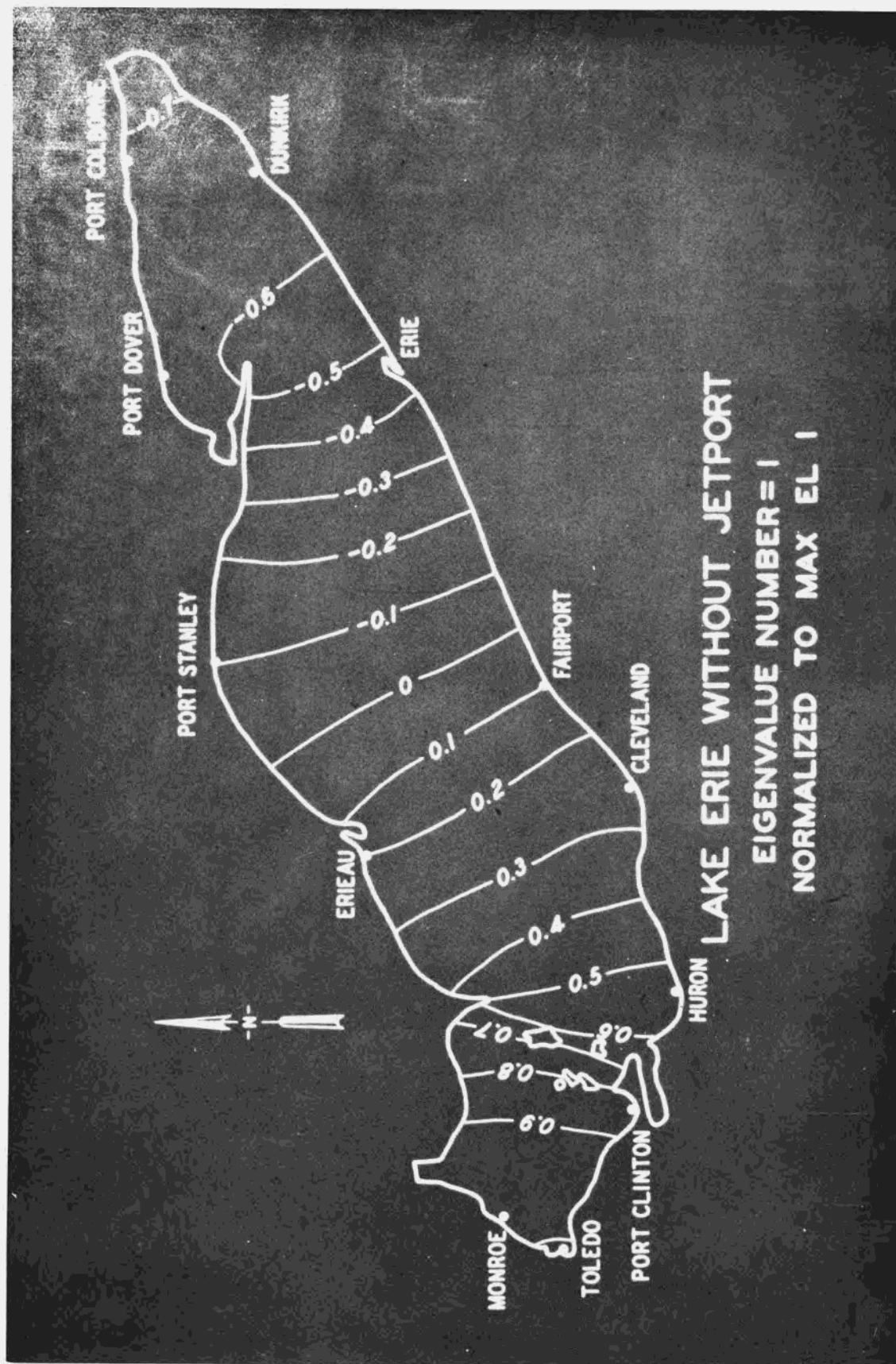


Fig. 15b. Fundamental mode of oscillation calculated by FEM

# SEICHE RELATIVE AMPLITUDE COMPARISON

<u>LOCATION</u>	<u>OBSERVED*</u>	<u>COMPUTED</u>
BUFFALO	-0.70	-0.72
PORT COLBORNE	-0.66	-0.69
DUNKIRK	-0.66	-0.66
PORT DOVER	-0.60	-0.63
ERIE	-0.50	-0.48
PORT STANLEY	-0.19	-0.09
FAIRPORT	0.13	0.09
ERIEAU	0.27	0.21
CLEVELAND	0.25	0.27
HURON	0.49	0.54
PORT CLINTON	0.52	0.86
TOLEDO	1.00	1.00
MONROE	0.95	0.99

\* FROM PLATZMAN AND RAO (1963)

included in the model. Various applications to actual inlets will be made within calendar year 1975.

- c. In addition to the models now being applied in the Lake Erie Jetport study, a time-dependent, rigid lid model including non-linear convection, horizontal diffusion, variable density, and coupling with the energy equation, will be operationalized at WES. Developed by Paul and Lick (1973), the model has been used to study the flow from the Cuyahoga River into Lake Erie under conditions such that the river water is either at the same temperature, warmer, or cooler than lake water. The model is presently being used by Paul and Lick to study the thermal efflux from power plants and is also being applied by Gedney, Molls and Paul to a study of the overall circulation in Lake Erie under stratified conditions.
- d. A wave information program is currently underway at WES and its primary objective is to produce a wave climate on the Great Lakes for application to the design of dredged material retaining structures. Specification of the wave climate will include: 1) probabilities of wave heights, wave periods, and approach directions at the edge of the breaker zone; 2) estimates of recurrence intervals for large waves; and 3) estimates of probable errors associated with hindcasted wave heights. Several numerical hindcast models will be tested for accuracy against recorded wave spectra around the Great Lakes. These models compute the entire five-dimensional energy spectrum of waves (  $f(x,y,\omega,g,t)$  ). With the numerical models, the effects of bottom topography on wave propagation and energy loss due to the interaction of waves with the bottom can be computed. Also included in the models are the effects of fetch length and width limitations and the effect of finite wind durations. The development of the models has been carried out primarily by groups of researchers at Scripps Institute of Oceanography, New York University and Westinghouse Ocean Research Laboratory. For some details of the concepts of these models, the reader is referred to Barnett (1966) and Inoue (1966).

10. SUMMARY. To recap, a broad spectrum of numerical models, based on long wave theory approximations, has been presented. These models are all presently being used and maintained at WES. A continuing effort to improve the models is also a major objective. WES has always played the major role in the United States in regards to physical models of hydraulic systems, whether they be harbors, inlets, rivers, spillways, etc. With numerical modeling now being recognized as an important tool in the investigation of hydraulic processes, it becomes imperative for WES to remain abreast of the state-of-the-art practices in such efforts. Throughout the descriptions of applications of the various models, the authors have tried

to stress the importance of using numerical models as a complementary activity when conducting hydraulic investigations.

An important consideration to keep in mind is that both numerical and physical models each have basic limitations and strengths. In particular, the assumptions and constraints inherent in each model must be cautiously considered in the application of numerical models of long-waves and wind-driven circulations. Both the selection of the prototype conditions being modeled as well as the interpretation of the results must be reasonably consistent with the model assumptions.

#### REFERENCES

1. Baker, B. B. and Copson, E. T., (1950), The Mathematical Theory of Huygen's Principle, Oxford University Press, London.
2. Barnett, T. P., (1966), "On the Generation, Dissipation, and Prediction of Ocean Wind Waves", PhD Dissertation, University of California, San Diego, 190 pp.
3. Desai, C. S. and Abel, J. F., (1972), Introduction to the Finite Element Method; A Numerical Method for Engineering Analysis, Van Nostrand Reinhold Co., New York, 1972, 477 pp.
4. Davidson, D. D. and Whalin, R. W., (1974), "Hydraulic Model Investigations and Velocity Predictions of Potential Landslides at Libby Dam and Lake Koocanusa," U. S. Army Engineer Waterways Experiment Station, Vicksburg, Miss., Technical Report H-74-15, December 1974.
5. Gedney, R. T. and Lick, W., (1972), "Wind-Driven Currents in Lake Erie," *Journal of Geophysical Research*, Vol. 77, No. 15.
6. Haq, A. and Lick, W., (1975), "On the Time-Dependent Flow in a Lake," *Journal of Geophysical Research*, Vol. 80, No. 3, January 20, 1975.
7. Hwang, LiSan, Butler, H. L., and Divoky, D. J., (1972), "Tsunami Model: Generation and Open-Sea Characteristics," *Bulletin of the Seismology Society of America*, Vol. 62, No. 6, Dec 1972.
8. Inoue, T., (1966), "On the Growth of the Spectrum of a Wind Generated Sea According to a Modified Miles-Phillips Mechanism and its Application to Wave Forecasting," New York University, School of Engineering and Science, Geophysical Science Laboratory, TR-67-5, 74 pp.

REFERENCE CONTINUED

9. Lee, Jiin-Jen, (1969), "Wave Induced Oscillations in Harbors of Arbitrary Shape," U. S. Army Engineer Waterways Experiment Station, Vicksburg, Miss., Contract Report H-69-2, Dec 1969.
10. Lee, Jiin-Jen and Raichlen, F., (1971), "Wave Induced Oscillations in Harbors with Connected Basins," U. S. Army Engineer Waterways Experiment Station, Vicksburg, Miss., Contract Report H-71-2, August 1971.
11. Leendertse, J. J., (1967), "Aspects of a Computational Model for Long-Period Water-Wave Propagation," The Rand Corporation, RM-5294-P4.
12. Paul, J. and Lick, W., (1973), "A Numerical Model for a Three-Dimensional, Variable Density Jet," Proceedings of the Sixteenth Conference on Great Lakes Research, Ann Arbor, Michigan, 1973.
13. Platzman, G. W., (1963), "The Dynamical Prediction of Wind Tides on Lake Erie," Meteorological Monographs, Vol. 4, No. 26, pp. 1-44.
14. Raney, D. C. and Butler, H. L., (1975), "A Numerical Model for Predicting the Effects of Landslide Generated Water Waves," U. S. Army Engineer Waterways Experiment Station, Vicksburg, Miss., Research Report H-75-1, February 1975.
15. Raney, D. C., Durham, D. L. and Butler, H. L., (1975), "Lake Erie International Jetport Numerical Circulation Study," to be published, Waterways Experiment Station.
16. Welander, P., (1957), "Wind Action on a Shallow Sea: Some Generalizations of Ekman's Theory," Tellus IX, Number 1.
17. Whalin, R. W., Garcia, A. W. and Butler, H. L., (1974), "Effect of Source Orientation and Location in the Aleutian Trench on Far-Field Tsunami Amplitude," presented at the Wellington Meeting of the Committee on Tsunamis, International Union of Geodesy and Geophysics, Victoria University of Wellington, Wellington, New Zealand, Jan 1974 (Proceedings to be published).



# NOTATIONS

$A$	Surface area of study region
$C$	Frictional coefficient
$C_d$	Drag coefficient for calculation of wind stress
$E_v$	Eddy diffusivity coefficient
$f$	Coriolis parameter
$g$	Acceleration of gravity
$H_0^{(1)}$	Zero order Hankel function of the first kind
$j, k$	Subscripts used to denote spatial coordinates
$K$	Wind stress coefficient in boundary conditions
$k$	Wave number
$L_x, L_y$	Landslide force components
$n$	Superscript used to denote a time instant
$n$	Refers to a normal line
$R_x, R_y$	Components of forcing terms in Landslide Model -- excluding forces due to slide
$r$	Distance function
$r_e$	Radius of earth
$S$	Distance
$t$	Time
$U, V$	Integrated velocity components
$u, v, w$	Three-dimensional particle velocity components
$v_x, v_y$	Components of landslide velocity
$W_a$	Wind speed
$x, y, z$	Cartesian coordinated

# NOTATIONS CONTINUED

$\alpha$	Viscous drag parameter
$\beta$	Pressure drag parameter
$\gamma_i$	Coefficients in stream function equation
$\Delta$	Interval operator
$\epsilon$	Ground motion parameter
$\eta$	Surface elevation
$\theta$	Latitudinal coordinate
$\lambda$	Eigenvalue
$\rho_a$	Density of air
$\vec{T}_B$	Bottom stress vector
$\vec{T}_w$	Wind stress vector
$\phi$	Longitudinal coordinate
$\chi$	Varitional integral
$\psi$	Stream function
$\omega$	Frequency

AUTOMATED DATA ACQUISITION AND CONTROL  
SYSTEMS FOR HYDRAULIC WAVE MODEL

Don L. Durham and Homer C. Greer, III  
U. S. Army Engineer Waterways Experiment Station  
P. O. Box 631, Vicksburg, Miss. 39180

**ABSTRACT.** Automated Data Acquisition and Control Systems (ADACS) have been designed and built at the Waterways Experiment Station for the purpose of collecting wave data and controlling operations of hydraulic wave models. The computer hardware configuration for each system consists of a minicomputer with 32K 16-bit words of memory, a magnetic tape controller with two 9 track tape drives, one moving head disk controller with one removable platter and one non-removable platter, an interval timer (1  $\mu$ sec), an analog to digital 12-bit converter featuring 64 analog ( $\pm 10$  volts) inputs and a 45 KHz multiplexer, an ASR 33 teletype unit, 96 sense/control lines and one matrix electrostatic printer/plotter. The ADACS are capable of automatically calibrating the wave sensors, controlling wave generators, acquiring data from the sensors at a high sampling rate, and analyzing test data. Data are taken and recorded on disc or magnetic tape for direct analyses by the minicomputer system or on magnetic tape in a format compatible with a Honeywell G635 for backup analyses. Automatic calibration of wave sensors has reduced the time required to calibrate the sensors by a factor of four. In addition, at least twice the number of tests can be run during a day with test results analyzed at completion of model tests by minicomputer system or within 24 hours by backup procedures for Honeywell G635.

**1. INTRODUCTION.** The Wave Dynamics Division (WDD) of Hydraulics Laboratory is primarily concerned with problems involving wave phenomena in harbors, tidal inlets, and on the open coast and the effect of these phenomena on coastal structures (breakwaters, jetties, groins, bulkheads, etc.), harbor oscillations, moored ship response, harbor circulation and flushing, navigation conditions thru tidal inlets and harbor entrances, maintenance of navigation channels, and sediment transport. Most WDD studies involve the use of both physical and numerical hydraulic wave models.

The increase in the number and complexity of wave model studies conducted each year and the use of such models to solve long wave problems (requiring large model areas) have vividly demonstrated a need for automation of modeling procedures including operation, data collection, and data analysis. The physical size of models (Fig. 1) involving the study of long wave phenomena, the vast complex of basins and channels requiring detailed study, and the large number of tests necessary to evaluate the effect of improvement plans on harbor circulation and oscillation eliminate manual

---

This paper was presented at the 1975 Army Numerical Analysis and Computers Conference.

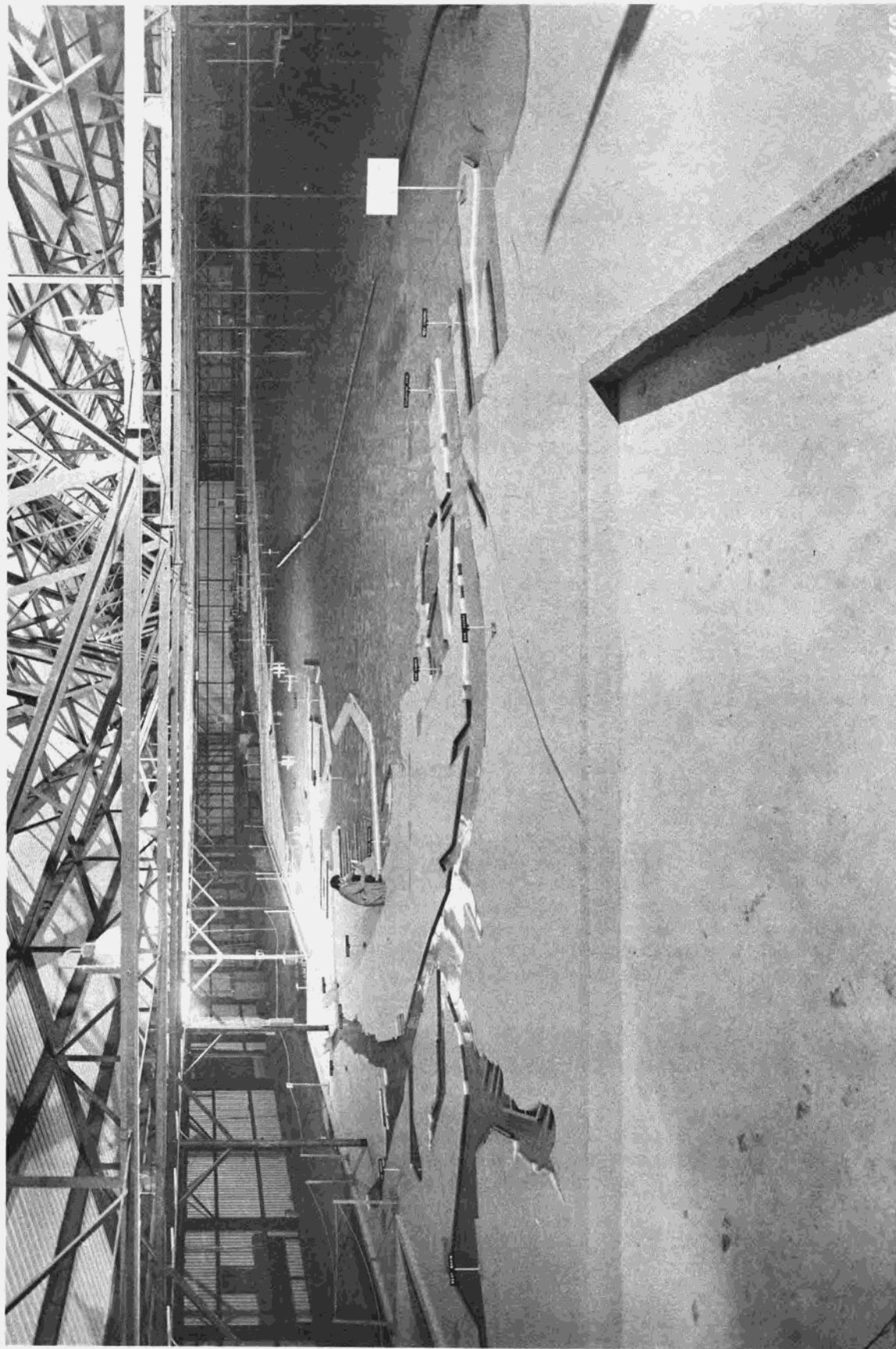


Fig. 1. Hydraulic wave model of Los Angeles-Long Beach Harbors

procedures for data collection and analysis. For example, the Los Angeles and Long Beach Harbors model (Whalin et al, 1974) covers an area of 45,930 square feet. Considerable effort over the past several years has been devoted to:

- a. Design and development of automated systems for acquisition of wave data and for model controls.
- b. The automation of data reduction and analyses.

There are presently three automated systems for hydraulic models in the Hydraulics Laboratory. One system, (Daggett, 1971) developed a few years ago, is operational on the New York Harbor model in the Estuaries Division of the Hydraulics Laboratory. The other two systems, which are the subject of this paper, are operational on various wave models in WDD. The need for automated model procedures and the success of the above automated systems have led to future plans for providing model automation to most modeling facilities of the Hydraulics Laboratory. Such plans are basically to extend the automated capabilities of the above systems to other facilities and/or installation of additional automated systems.

2. SYSTEM CONFIGURATION. The automated systems for wave models have been given the title "Automated Data Acquisition and Control System" (whose acronym is ADACS) and have two primary functions: (1) automated acquisition of wave data in a format (magnetic tape or disc) compatible for computer reduction and analyses and (2) automated control of wave sensor calibrations and of the wave generators. The system configuration (Fig. 2) of ADACS consists of the following subsystems:

- a. Digital Data recording and controls.
- b. Analog recorders and channel selection circuits.
- c. Wave sensors and interfacing equipment.
- d. Wave generators and control equipment.

The digital data recording and control subsystem is basically a minicomputer (one  $\mu$ sec memory cycle time) with the following characteristics and peripheral devices:

- a. 32K, 16 bit words of core memory.
- b. Analog to digital pack featuring 64 analog inputs ( $\pm 10$  volts, FS), 45 KHz multiplexer and 12 bit (including sign) A/D converter.
- c. Interval timer with a one  $\mu$ sec counter.



- d. A long line adapter and a universal controller.
- e. Two direct memory access channels.
- f. Moving head disk with one removable platter and one non-removable platter (1.1 million word storage capacity, 100 Kc transfer rate).
- g. Magnetic tape controller with two 9-track magnetic tape units (25 ips, 800 bpi, 10 Kc transfer rate).
- h. Teletype unit with keyboard/printer and 10 cps paper tape reader/punch.
- i. 96 sense and control lines.
- j. Matrix electrostatic printer/plotter (print: 1620 lines/min, 132 chars/line and 128 char set; plot: 4 in/sec and 100 styli/in).

The analog recording subsystem acts as a backup for ADACS and a visual display for operator inspection of analog signals from wave sensors. This subsystem has manual or automated selection and control of five 12-channel oscillographs.

The wave sensor subsystem includes the following major components:

- a. Fifty wave height sensors and stands,
- b. Fifty channels of signal conditioning equipment,
- c. Power supplies, and
- d. Manual and automatic calibration equipment.

The last subsystem, wave generators and control, is not completely implemented in the present configuration of ADACS. Upon completion (~6 months) the wave generators will utilize electrohydraulic actuators for driving a wave board instead of mechanically gear-driven wave boards. Presently, start/stop controls are available for mechanical generators; however, controls for wave period and amplitude must be set manually. Except for the wave sensors and wave generators, all subsystems (Fig. 3) of each ADACS are compactly housed in a trailer to provide easy control of the environment of each complete system.

3. WAVE SENSORS. The data acquired from wave models are the water surface variations about a reference water level. This information is collected at selected geographic locations within the model for specified wave conditions at the wave generator. Wave sensors are used to obtain

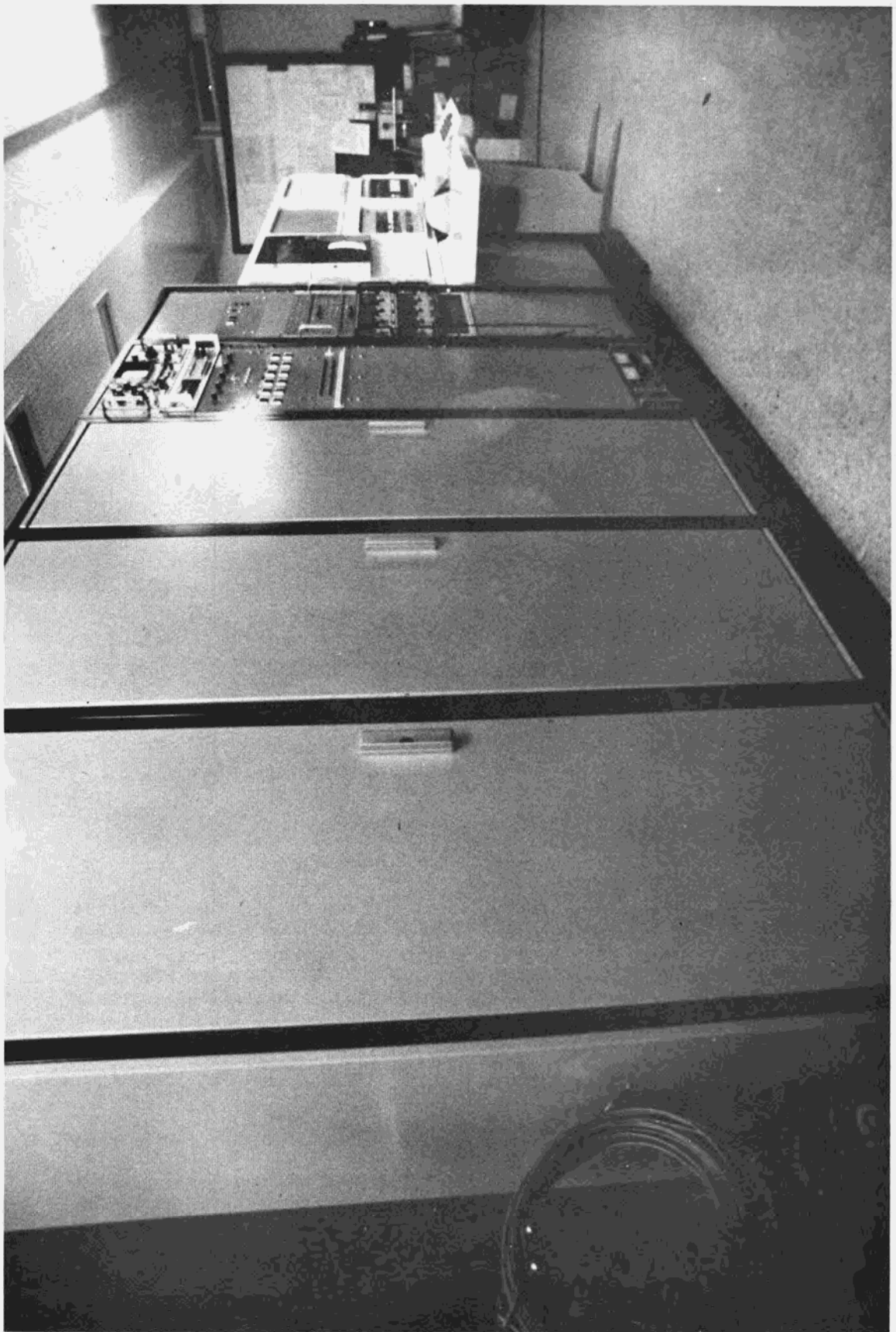


Fig. 3. Automated data acquisition and control system



this information at selected locations in the model. Several types of model wave sensors are employed at WES to obtain water surface data. The difference between these various wave sensors is basically the physical phenomena (pressure, conductivity or resistance, capacitance, and acoustic reflections) which the sensor measures and relates to a surface displacement. WES is presently conducting an evaluation program for both wave (long and short periods) and velocity sensors. Results of this program should be completed in the near future. The discussion in this presentation will be limited to the water-surface-piercing, parallel-rod gages (Fig. 4).

Each water-surface-piercing, parallel rod wave sensor is connected to a wheatstone bridge (Fig. 5). The transducer measures the conductance of the water between two parallel rods mounted vertically. The conductance is directly proportional to the depth of submergence of the two rods in the water. The output of each wave sensor is routed through shielded cables to its signal conditioning equipment where it is processed for recording. The signal conditioning equipment consists of a carrier amplifier and the various power supplies. This system can detect changes in water surface elevations to an accuracy of 0.001 of a foot. To obtain this accuracy, ultra-stable power supplies and better-than-average signal to noise ratios are necessary. The carrier source for the wave sensor bridge maintains a variation of less than 0.025%. The noise in the system is less than +5 millivolts for +10 volts full range. Thus the signal to noise ratio is -66dB. To maintain this signal to noise ratio, it is necessary to use shielded cables wherever possible, to discriminate selections of grounds, to use high quality components, and to have a good maintenance schedule.

The output of the signal conditioning equipment is connected through shielded cables to analog oscillographs where an analog time history is recorded and to the analog multiplexer of the digital recording subsystem where it is digitized and recorded in a binary format on magnetic tape and/or disc.

In order to convert the water elevation data in millivolts to water surface elevations in feet, each wave sensor must be calibrated. The capability of automatically calibrating the wave sensors (maximum of 25 rods simultaneously) prior to collecting data is provided by ADACS. Thus, ADACS performs in two modes: (1) calibration and (2) acquisition. For both modes of operation, ADACS has software and manual control capabilities.

**4. CALIBRATION.** In order to calibrate each set of parallel rods, the voltage from the signal conditioning equipment is monitored and recorded as the parallel rods are moved vertically a known distance into or out of the water. A precision, linear-position potentiometer is located on the wave sensor stand and is coupled directly to the parallel rods by a gear train driven by electric motor. By moving vertically the coupled wave sensor and potentiometer wiper with the electric motor and by monitoring the output voltage from the potentiometer, the wave sensor can

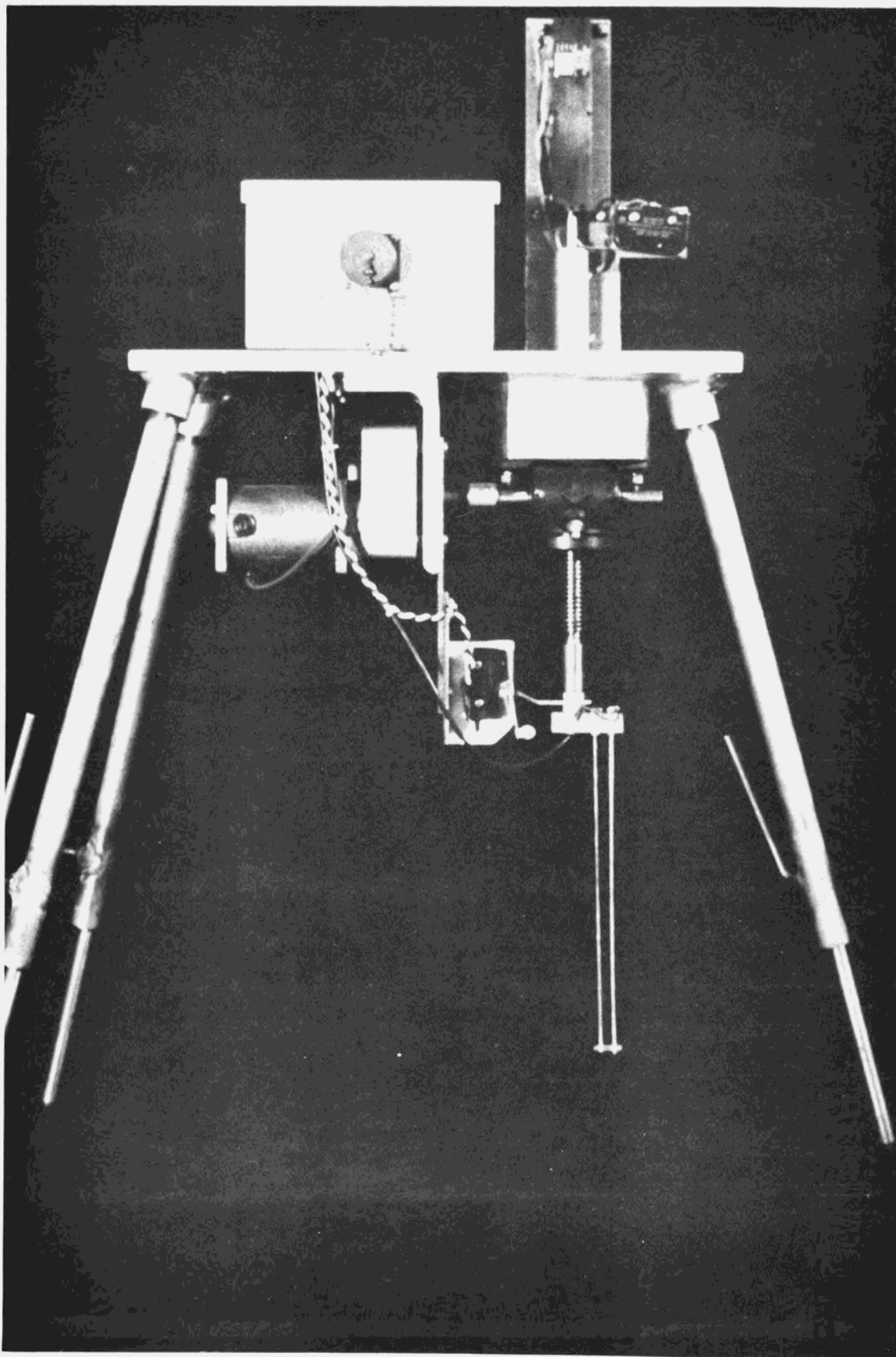
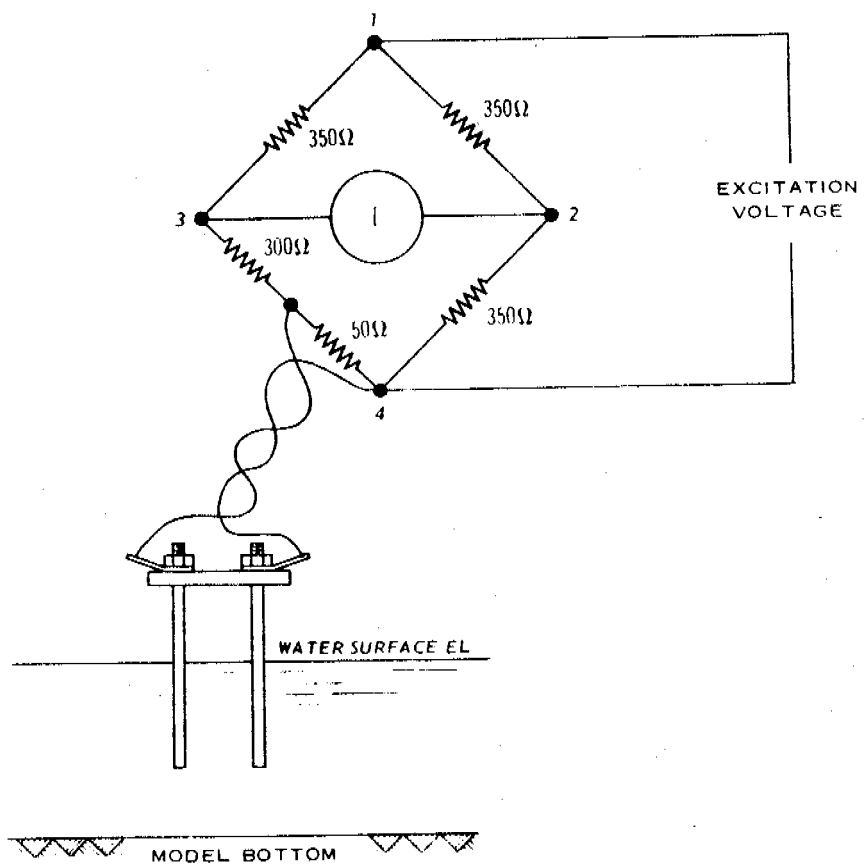


Fig. 4. Parallel-rod wave sensor



BRIDGE TRANSDUCER FOR  
PARALLEL-ROD WAVE SENSOR

Fig. 5. Schematic of parallel-rod bridge transducer

be moved vertically a precise distance. The electric motor for each wave sensor is controlled by a control/sense line and a relay contact. The minicomputer controls the vertical movement of each wave sensor by actuating the control/sense line. The central processing unit (cpu) acts as a voltage comparator by monitoring the potentiometer voltage and comparing it to a reference voltage which is determined from desired displacement and potentiometer calibration. When the voltage comparison is satisfied, the control/sense line is reactivated; the electric motor stops; and voltage samples from the rods and potentiometers are acquired. By systematically moving each wave sensor through 11 quasi-equally spaced locations over the range of the rod length used, voltage versus known displacements are obtained from which a calibration curve for each sensor can be calculated and recorded on magnetic tape or disc. After collecting the calibration data, the minicomputer analyzes these data by least squares fitting a set of curves (linear, quadratic, or spline) to the data, determining the best order of fit, and comparing the maximum deviation of the best fit to a previously acceptable value for this maximum deviation. If the fitted curves are not acceptable, the minicomputer flags that channel in the calibration record on the magnetic tape or disc for further analyses. Any malfunctioning sensors are listed on the teletype for the operator to determine the required action (i.e., accept present calibration; clean bad rods, recalibrate, etc.). Having completed the calibration mode, the original calibration data for each set of parallel rods and potentiometer as well as the calibration coefficients are written into a file on magnetic tape or disc. Analytical considerations of calibration curve fitting, accuracy of reference potentiometers and repeatability test of calibration procedures have demonstrated that an accuracy of  $\pm 0.001$  feet in wave height is obtainable.

5. DATA ACQUISITION. During the acquisition mode wave data for a specified wave condition at the wave generator are collected from a maximum of 50 wave sensors, recorded on analog strip charts, digitized and recorded on magnetic tape or disc for further analyses. The sampling scheme is quite flexible and can be tailored for different applications with maximum thru-put rates theoretically limited by the multiplexer rate (45KHz) and allocatable buffer size. However, for specific types of tests the execution time of applications software and the specified number of discrete samples per wave period degrade the about maximum thru-put rate. The present sampling scheme used for each wave sensor is 60 discrete voltage samples equally spaced over each wave period for a predetermined number of periods (normally 24). The minicomputer calculates from input parameters the required timing interval between multiplexer scans to provide the correct sampling rate and initializes counters for determining completion of wave tests. In addition, it controls the starting of the wave generators, lags the beginning of data acquisition by a specified number of wave periods after starting the generators, and provides timing pulses for synchronizing and controlling the analog recorders. At completion of the acquisition mode, the calibration data and wave data have been recorded in both analog and digital form. In binary form on magnetic

tape or disc, these data with a header for test identification and pertinent parameters are available from disc for direct analyses by the minicomputer or on magnetic tape in a format suitable for backup analyses on WES Honeywell G635 computer.

6. DATA ANALYSIS. The analysis of the data may be performed by either the minicomputer or by a Honeywell G635 of the Automated Data Processing Center at WES. Schematically, the procedure for data analyses is as follows:

a. Program Initialization

- (1) Input test parameters and option flags.
- (2) Read and decode data tape or disc file.
- (3) Demultiplex data files.

b. Wave Record Analyses

- (1)  $H \pm \sigma_H$  and  $\bar{T} \pm \sigma_T$ .
- (2)  $H_{RMS}$ .
- (3)  $H_x$  where  $x$  is a specified percent of the highest wave heights, normally  $x = .333$ ; thus  $H_{1/3}$  is significant wave height.
- (4) Option to plot wave heights versus time.

c. Fourier Analyses

- (1) Autospectrum
  - (a) Amplitude-frequency
  - (b) Energy
  - (c) Spectral smoothing
  - (d) Plots of above parameters
- (2) Cross spectrum
  - (a) Energy
  - (b) Coherency
  - (c) Phase
  - (d) Plots of above parameters

The results of the data analyses can be plotted by pen plotter or CRT plotter (microfilm and hard copy) on Honeywell G635 system and by electrostatic printer on minicomputer. For each model study, the original data and analyzed results of all tests are permanently stored on magnetic tape for future reference. With these analyses of wave records from various harbor locations, the amplification factors or harbor response for various input wave conditions are estimated. These results provide the hydraulic engineers with the basic data whereby the effects of various proposed expansions and modifications to an existing harbor can be evaluated.

7. SUMMARY. Automated control of the calibration of wave sensors has reduced the time required to calibrate sensors by a factor of four. In addition, at least twice the number of model tests can be run during a day. The cost for data analyses has decreased by at least a factor of two with test results returned to the project engineer immediately after model testing when analyzed by minicomputer subsystem and over night when analyzed by backup mode with Honeywell G635 system. Thus, the automated procedures described in this paper have enhanced the modeling capabilities and increased the efficiency of modeling procedures. Future efforts include an evaluation program for wave and tidal height sensors, velocity meters, spectral wave generation, and an expansion of automated capabilities to other model facilities.

#### REFERENCES

1. Daggett, Larry, "Automation of Control, Data Acquisition and Data Processing for the New York Harbor Model." Paper presented ASCE Hydraulics Specialty Conference, Iowa City, Iowa, Aug 1971.
2. Whalin, R. W., Chatham, C. E., Durham, D. L., and Pickett, E. B., A Case History of Los Angeles-Long Beach Harbors, ASCE Proc. of International Symposium on Ocean Wave Measurement and Analysis, Vol 1, Sept 1974.

ATTENDEES

1976 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE

11-12 February 1976

US Army Research Office

Research Triangle Park, North Carolina

Mr. Donald C. Adams  
USA Air Mobility R&D Lab  
Moffett Field, CA

Mr. William S. Agee  
USA White Sands Missile Range  
White Sands Missile Range, NM

Mr. John C. Ahlquist  
Waterways Experiment Station  
Vicksburg, MS 39180

Mr. William Barnhart  
USA Test & Evaluation Command  
Aberdeen Proving Ground, MD 21005

Mr. Carl B. Bates  
USA Concepts Analysis Agency  
Bethesda, MD

Mr. James R. Batts  
Bell Aerospace Company  
Buffalo, NY

Dr. Vitalius Benokraitis  
USA Ballistic Research Labs  
Aberdeen Proving Ground, MD

Mr. Deane B. Blazie  
USA Human Engineering Lab  
Aberdeen Proving Ground, MD

Dr. Paul Boggs  
US Army Research Office  
Research Triangle Park, NC

Mr. Alfred Brandstein  
Harry Diamond Laboratories  
Adelphi, MD 20783

Dr. Achi Brandt  
IBM Thomas J. Watson  
Research Center  
Yorktown Heights, NY

Mr. H. Lee Butler  
USA Engineer Waterways Experiment  
Station  
Vicksburg, MS

Dr. Jagdish Chandra  
US Army Research Office  
Research Triangle Park, NC

Mr. Raymond F. Coakley  
USA Mobility Equipment R&D Command  
Ft. Belvoir, VA

Ms. Nancy J. Cobean  
USA Natick Development R&D Command  
Natick, MA

Dr. Philip C. Cooley  
Research Triangle Institute  
Research Triangle Park, NC 27709

Dr. Fred Crary  
University of Wisconsin-Madison (MRC)  
Madison, WI

Mr. E. David Crockett  
Hewlett-Packard  
Cupertino, CA

Mr. Thomas Dames  
USA Electronics Command  
Ft. Monmouth, NJ 07703

Professor Carl de Boer  
University of Wisconsin-Madison (MRC)  
Madison, WI

Professor E. J. Desautels  
University of Wisconsin-Madison  
Madison, WI

Dr. Francis Dressel  
US Army Research Office  
Research Triangle Park, NC

Mr. Don L. Durham USA Engineer Waterways Experiment Station Vicksburg, MS	Mr. Willard M. Holmes USA Missile Command Redstone Arsenal, AL
Dr. Russell Eaton, III USA Mobility Equipment R&D Center Ft. Belvoir, VA	Dr. J. C. Ingram Harry Diamond Laboratories Adelphi, MD
Mr. Sylvan H. Eisman USA Frankford Arsenal Philadelphia, PA	Mr. Robert I. Isakower Picatinny Arsenal Dover, NJ
Professor David J. Farber University of California Irvine, CA 92665	Mr. Van R. Jones USA Materiel Systems Analysis Activity Aberdeen Proving Ground, MD
Dr. A. S. Galbraith Durham, NC	Mr. Robert E. Kasten Rock Island Arsenal Rock Island, IL
Professor Thomas M. Gallie Duke University Durham, NC	Mr. S. Kravitz Aeronutronic Ford Corporation Palo Alto, CA
Mr. Edward Goldstein USA Test & Evaluation Command Aberdeen Proving Ground, MD	Mr. Abram Leff USA Mobility Equipment R&D Center Ft. Belvoir, VA
Professor Gene H. Golub Stanford University Stanford, CA	Mr. Truman L. Mabee Electronics Associates, Inc. Huntsville, AL
Mr. Charles E. Gray Aeronutronic Ford Corporation Palo Alto, CA	Professor Peter N. Marinos Duke University Durham, NC
Dr. Robert Todd Gregory University of Tennessee Knoxville, TN	Mr. C. Merritt, Jr. USA Natick R&D Command Natick, MA
Mr. David Grobstein Picatinny Arsenal Dover, NJ	COL Lothrop Mittenthal US Army Research Office Research Triangle Park, NC
Dr. Donald F. Haskell USA Ballistic Research Labs Aberdeen Proving Ground, MD	Dr. David Myers Research Triangle Institute Research Triangle Park, NC
Dr. Morton A. Hirschberg USA Ballistic Research Labs Aberdeen Proving Ground, MD	Dr. Thomas L. Nichols USA Natick R&D Command Natick, MA



Mr. David T. O'Brien  
Harris Corporation  
Rockville, MD

Professor Merrell Patrick  
Duke University  
Durham, NC

Professor Robert Plemmons  
University of Tennessee  
Knoxville, TN

Professor Louis B. Rall  
University of Wisconsin-Madison  
(MRC)  
Madison, WI

Mr. Donald C. Raney  
USA Engineer Waterways Experiment  
Station  
Vicksburg, MS

Dr. Robert P. Reklis  
USA Ballistic Research Labs  
Aberdeen Proving Ground, MD

Professor J. Barkley Rosser  
University of Wisconsin-Madison  
(MRC)  
Madison, WI

Mr. Paul K. Senter  
Waterways Experiment Station  
Vicksburg, MS

Dr. Babu Shah  
Research Triangle Institute  
Research Triangle Park, NC

Professor Philip W. Smith  
Texas A&M University  
College Station, TX

Mr. Royce Soanes  
Watervliet Arsenal  
Watervliet, NY

Mr. Larry Sturdivan  
Edgewood Arsenal  
Aberdeen Proving Ground, MD

Dr. Ivan Sutherland  
Rand Corporation  
Santa Monica, CA

Dr. Stan Taylor  
Ballistic Research Laboratories  
Aberdeen Proving Ground, MD

Mr. Vincent K. Taylor  
Canadian Defence Liaison Staff  
Washington, DC 20008

1LT Steven G. Thompson  
USA Mobility Equipment R&D Center  
Ft. Belvoir, VA

Mr. R. H. Thornton  
Research Triangle Institute  
Research Triangle Park, NC

Mr. Yuet Tsui  
Duke University  
Durham, NC

Dr. Ronald P. Uhlig  
USADARCOM  
Alexandria, VA

Professor Senol Utku  
Duke University  
Durham, NC

Mr. Robert Voight  
NASA-Langley Research Center  
Hampton, VA

Mr. M. Waller  
USAOTEA  
Falls Church, VA

Mr. Alan Weinberger  
USA Mobility Equipment R&D Center  
Ft. Belvoir, VA

Mr. William P. White, Jr.  
USA Air Mobility R&D Laboratory  
Hampton, VA

Dr. Leland Williams  
Triangle University Computation Center  
Research Triangle Park, NC

LTC Marion C. Winebarger  
United States Military Academy  
West Point, NY

Dr. Ralph A. Willoughby  
IBM Research  
Yorktown Heights, NY

Mr. Eric Wolf  
Bolt Beranek Newman  
Arlington, VA

Professor J. M. Yohe  
University of Wisconsin-Madison  
(MRC)  
Madison, WI

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ARO Report 76-3	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PROCEEDINGS OF THE 1976 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE		5. TYPE OF REPORT & PERIOD COVERED Interim Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Army Mathematics Steering Committee on behalf of the Chief of Research, Development and Acquisition		12. REPORT DATE September 1976
		13. NUMBER OF PAGES 535
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) JS Army Research Office DRXRO PO Box 12211 Research Triangle Park, NC 27709		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited. The findings in this report are not to be construed as an Official Department of the Army position, unless so designated by other authorized documents.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES This is a technical report resulting from the 1976 Army Numerical Analysis and Computers Conference. It contains papers on computer aided design and engineer- ing, as well as papers on numerical analysis.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <div style="display: flex; justify-content: space-between;"> <div style="width: 48%;"> transportability of mathematical algorithms high-speed computing on a mini-computer distributed computer ring networks linearized least squares nonlinear spline regression rootfinding methods optimizing symmetric successive overrelaxation VP splines the transonic potential equation utilizing real-time test data shaped charged projectile firepower- kill a mine detection system </div> <div style="width: 48%;"> hydraulic models of tidal inlets optimal instrumentation planning the Buckingham Pi theorem graphic program for determining mass properties of planar solids testing algorithms for a mini-computer on a maxi processing of mass spectral data using a mini-computer recursive digital filtering simulating flow in an inlet-wetlands system landslide-generated water waves Euler-Maclaurin integration cancellation and rounding errors </div> </div>		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)